SOLAR SYSTEM MODELER: A DISTRIBUTED, VIRTUAL ENVIRONMENT FOR SPACE VISUALIZATION AND GPS NAVIGATION

THESIS

Gary E. Williams
Captain, USAF

AFIT/GCS/ENG/96D-29

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/96D-29

SOLAR SYSTEM MODELER: A DISTRIBUTED, VIRTUAL ENVIRONMENT FOR SPACE VISUALIZATION AND GPS NAVIGATION

THESIS
Gary E. Williams
Captain, USAF

AFIT/GCS/ENG/96D-29

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

AFIT/GCS/ENG/96D-29

SOLAR SYSTEM MODELER: A DISTRIBUTED, VIRTUAL ENVIRONMENT FOR SPACE VISUALIZATION AND GPS NAVIGATION

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Systems

Gary E. Williams, B.S.

Captain, USAF

December, 1996

Acknowledgments

AFIT has truly been one of the *once in a lifetime* opportunities. I would like to thank God most of all for sustaining me throughout the program. I also thank my wife and children for their support and tolerance throughout the last eighteen months. I knew I could always count on her encouragement on those occasions I would rather forget. The AFIT faculty provided outstanding guidance and assistance, particularly in a few key areas where I had little of the knowledge required for this thesis. Specifically, I would like to thank Lt Col Jack Kloeber for his guidance on statistics and random variate generation, Dr. William Wiesel for explaining the finer points of astrodynamics, and Capt Stewart DeVilbiss for his insight on the GPS. I would also like thank my fellow students for their friendship, the best part of the AFIT experience. Finally, I thank my advisor, Lt Col Martin Stytz, for his vision and giving me the opportunity to work in a superb lab on an extremely interesting project.

Gary E. Williams

Table of Contents

List of Figures

List of Tables

ABSTRACT

The *Solar System Modeler* (SM) extends the *Space Modeler* developed in 1994. It provides a virtual environment enabling an explorer to dynamically investigate near earth satellites, deep space probes, planets, moons, and other celestial phenomena. The explorer navigates the virtual environment via mouse selected options from menu panels while wearing a tracked, head mounted display (HMD). Alternatively, a monitor may replace the HMD and keyboard controls replace head tracking. The SM's functionality is extended by the ability to broadcast simulated GPS satellite transmissions in compliance with Distributed Interactive Simulation (DIS) protocol standards. The transmissions include information found in true GPS broadcasts that is required for a receiver to determine its location. The Virtual GPS Receiver (VGPSR) receives the GPS transmissions from the SM and computes the receiver's position with a realistic error based on numerous variables simulating those encountered in the real GPS system. The VGPSR is designed as a plug-in module for simulations requiring virtual navigation. The receiver's client application provides the VGPSR with the simulation time and the true position of the receiver. In return, the application receives a GPS indicated position.

SOLAR SYSTEM MODELER: A DISTRIBUTED, VIRTUAL ENVIRONMENT FOR SPACE VISUALIZATION AND GPS NAVIGATION

I. Introduction

1.1 Background

Virtual environments have been evolving since the 1960's when Ivan Sutherland developed the first Head Mounted Display (HMD) (13). Technology's remarkable advance has produced systems that now give us the required graphical displays, speed, memory, connectivity, and software to model, visualize and simulate highly complex environments (1, 22, 24, 32). These systems have application in many aspects of our culture including education, research, training, and entertainment (8, 33, 35). The United States Air Force, as well as other branches within the Department of Defense, has recognized the potential of such systems for training, system development, testing, and force assessment. This potential can be realized by simulating weapon systems prior to acquisition and analyzing battlefield tactics and their effectiveness through sophisticated wargames. Several initiatives have been launched over the last few years in support of these tasks. The technical communities involved in these activities have come to realize the challenge of accomplishing these goals.

The simulation community, including AFIT's Virtual Environments, 3D Medical Imaging and Computer Graphics Lab, has been researching large scale, distributed simulation for several years. Communication and effective interaction between distributed, heterogeneous simulations is currently an area of intense research, particularly within the Department of Defense. Such research has already resulted in the development of standards such as the

Distributed Interactive Simulation (DIS) protocol (39). DIS provides a foundation upon which developers build simulations with the ability to run autonomously in the local environment as well as participate in large-scale distributed exercises.

This is the third research project undertaken to develop large scale space distributed virtual environments. The first was Andrea Kunz's *Satellite Modeler* in 1993. Her effort focused on developing a virtual, near Earth space environment for visualizing and interacting with satellites and constellations of satellites in accurate orbits. The *Satellite Modeler* also had the capability to act as a player in DIS compliant distributed simulations (22). This work laid the foundation for the *Space Modeler* (SM) developed by Capt John Vanderburgh in 1994.

The *Space Modeler* extended the capability of the *Satellite Modeler* by adding a three-dimensional user interface and expanding the environment to include the entire solar system. The new interface gives the immersed user the ability to move freely within the environment with five degrees of freedom, control the speed of the simulation, and attach to objects within the environment (42). The expanded environment includes the Sun and nine planets as well as several moons, comets, and asteroids. Like its predecessor, the *Space Modeler* has the ability to participate in DIS compliant distributed simulations (42).

1.2 Problem Statement

There are two main thrusts to the research reported here. One area is the visualization of deep space probes such as Galileo and the Voyager and Pioneer series. The computations required to determine the location of a satellite are well known to scientists in the astronomical community. However, it is difficult for the layman to visualize an orbit given the mathematics alone. Extending the SM to include the orbits of deep space probes would enable a user to visualize a probe in the celestial environment providing insight to its location, motion, and relationships to other heavenly bodies.

The second thrust is the development of a GPS navigation capability for distributed virtual environments (DVEs). There are several components to this effort. First, the orbital information for the Global Positioning System (GPS) satellites needs to be updated to increase the accuracy of the orbits. Second, the GPS constellation must be expanded to include all GPS satellites currently in orbit. With the improved accuracy and completed constellation, the SM could be used in a distributed environment to simulate the GPS satellite broadcasts used by thousands of Department of Defense assets today.

The emphasis of this thesis is on the simulation capabilities of the SM. This thesis extends the capability of the SM in each of the problem areas presented.

1.3 Assumptions

This thesis is based on work accomplished by the developers of the *Satellite Modeler* and the *Space Modeler*. Consequently, I assumed the accuracy of the algorithms and implementations resulting from those efforts was correct and sufficient for my research. Because the accuracy of the orbits of the satellites and certain moons and planets was essential to the success of this research and since orbits change over time, I assumed the orbital data upon which the previous work was based required verification. This was a particularly important consideration for the GPS satellites that provide position information to other entities in distributed simulations. As a result, the orbital parameters of those satellites was updated as necessary.

Many of the assumptions made in the *Satellite Modeler* and *Space Modeler* was made for this work as well. These assumptions included the following (22, 42):

- A simplifying assumption that the earth is a sphere rather than oblate (this assumption would hold for the *Solar System Modeler*, but not the Virtual GPS Receiver).
- The software would be implemented using the current standard protocols (DIS) for distributed simulation.

- The software would make use of the *ObjectSim* application development framework, *Performer* graphics libraries, and C++.
- All software would execute on the Silicon Graphics Onyx graphics computer running the *IRIX* operating system.

1.4 Approach

This effort took a different approach than the typical sequence followed for AFIT theses. Here, multiple phases were conducted simultaneously to ensure functional software was available by the deadlines required for publications, conferences, and the sponsor.

First, a thorough review of the existing software from the *Space Modeler* was required to understand the work upon which this thesis is based. As a part of this review, I read both Kunz's and Vanderburgh's theses. Following this review, I studied astrodynamics principles in sufficient depth to understand the concepts and equations required to develop and implement the algorithms for propagating the orbits of the deep space probes. Development of the three dimensional models representing the deep space probes and the software to propagate their orbits and render them in the virtual environment followed.

Research for the GPS satellite extensions commenced after the deep space probes had been successfully added. First, I conducted a literature search to gain an understanding of the function and operation of the GPS satellites and to locate updated orbital data. This literature search was followed by completion of the GPS satellite constellation. Once all of the GPS satellites were correctly orbiting the Earth, I studied the DIS standard and extended the software to provide the capability to broadcast the desired navigational and time information for all satellites. After the satellites had broadcast capability, I developed a Virtual GPS Receiver to receive and decode the satellite broadcasts. This receiver is separate from the *Solar System*

Modeler and is intended to be integrated into client applications to provide virtual GPS navigation.

At the end of both the deep space probe and GPS satellite phases, verification and validation of the results was conducted to ensure the resulting software does what was expected and the orbits of new celestial objects are accurate.

Concurrent with the previously mentioned phases of this research, I conducted a comprehensive literature search in the areas of astrodynamics, GPS satellite navigation, distributed simulation, and virtual reality.

1.5 Thesis Presentation

This thesis is divided into six chapters, 2 appendices, a glossary of terms, and a bibliography. Chapter 2 provides background in the areas the reader is not assumed to have knowledge but that are necessary to understand this work. Chapter 3 identifies the specific requirements upon which this work is based. Chapter 4 presents the design and implementation of the software written to fulfill the requirements of Chapter 3. Chapter 5 presents the results of the work and Chapter 6 summarizes the conclusions and recommended areas of future research. The glossary of terms is provided to avoid needless digression in the presentation of concepts. Words appearing in the glossary are identified in the text by bold print.

1.6 Summary

This chapter has provided the foundation for my research. The background, problem statement, assumptions, and approach have been discussed. Furthermore, the presentation format of this document was presented. In the next chapter, background domain information is presented to provide the unfamiliar reader with the insight necessary to understand this research.

II. Background

This chapter provides background information concerning the underlying concepts employed in the development of the *Solar System Modeler.* It addresses virtual environments, fundamental astrodynamics, advanced distributed simulation, and satellite navigation using the Global Positioning System (GPS).

2.1 Virtual Environments

The origins of virtual environments (VE) can be traced back to the 1960's and Ivan Sutherland's work with head-mounted stereo displays. Probably the most notable source of VE technology has come from work associated with the development of vehicle simulators, especially aircraft (25). Many of these simulators were very expensive. However, advances in technology have brought the costs down to the point where a virtual environment can be developed for about the cost of an expensive workstation. As a result, we have seen the development of personal simulators with wide application. Real time interactive three-dimensional displays with frame rates greater than 20 frames per second have been realized through these advances.

Virtual environments provide an excellent forum in which to explore celestial activity ranging from the phases of Jupiter's moons to the orbits of man-made satellites, such as the GPS constellation. Due to the vastness of space, complex three-dimensional movement, such as the Shoemaker-Levy 9 collision with Jupiter, is very difficult to visualize in two dimensions. On the other hand, a three-dimensional representation of the same movement provides significantly more insight. This is evident in (40) when the two-dimensional diagrams of the comet's orbit are compared to the three-dimensional computer animation of the same event. A VE provides even greater enlightenment by allowing the user to interactively explore the environment. While most space visualization environments are designed to simulate a first person presence in space, they

have also been used to train ground-based operators and simulate mission planning and launch activities (42, 36). One such example, the Remote Access Virtual Environment Network (RAVEN), was used by NASA to assist in training astronauts for the highly successful repair of the Hubble Space Telescope in December 1993 (8).

The predecessors of the *Solar System Modeler*, the *Satellite Modeler* and the *Space Modeler*, laid a solid foundation by providing a virtual environment that accurately portrays the orbits of the Sun, planets, moons, comets and asteroids as well as near earth and deep space satellites. The orbits are determined using fundamentals of astrodynamics and the NORAD SGP4 satellite orbital model (22, 42). Furthermore, the system provides the capability to broadcast satellite position and attitude in real-time via the Distributed Interactive Simulation (DIS) network protocol (22, 42). The reader is encouraged to refer to (22) and (42) for additional information on the development of the virtual environment inherited by the *Solar System Modeler*.

Useful visualization of the space environment requires accurate modeling of orbital motion. An understanding of the fundamentals of astrodynamics is required for such modeling.

2.2 Fundamental Astrodynamics

2.2.1 The Fundamental Laws. Astrodynamics is the study of all aspects of the motion of natural and man-made objects in space. Its roots are traditionally traced to Tycho Brahe and Johann Kepler. Brahe is credited with the meticulous collection and recording of accurate data on the positions of the planets. However, he lacked the insight to realize the potential of his work. On the other hand, Kepler was unable to make such precise observations but diligently studied Brahe's data and succeeded where he fell short. In 1609, Kepler published the first two laws of planetary motion. The third law followed ten years later. Kepler's laws provided an accurate description of planetary motion for the first time (5:2). Kepler's laws are:

1. The orbit of each planet is an ellipse, with the Sun at a focus.
2. The line joining the planet to the sun sweeps out equal areas in equal times.
3. The square of the period of a planet is proportional to the cube of its mean distance from the Sun.

Approximately 50 years later, Isaac Newton discovered why the planets move as Kepler described. His work was finally published in *Principia* over 20 years later and introduced his three laws of motion (5):

1. Every body continues in its state of rest or uniform motion in a straight line unless it is compelled to change that state by forces impressed upon it.
2. The rate of change of momentum is proportional to the force impressed and is in the same direction as that force.
3. To every action there is always opposed an equal reaction.

Along with stating the laws of motion, *Principia* put forth two equations that are the basis for the equation of motion for planets and satellites. The first equation is based on Newton's second law of motion and the second equation is Newton's Law of Gravity. The Law of Gravity states that any two bodies attract one another with a force proportional to the product of their masses and inversely proportional to the square of the distance between them.

$$\frac{d(m_i v_i)}{dt} = F \qquad \text{(Newton's second law of motion)} \qquad (1)$$

$$\frac{GMm}{r^2} = F \qquad \text{(Newton's law of gravity)} \qquad (2)$$

Using these equations, the motion of one body within a system of bodies may be analyzed. This is commonly referred to as the n-body problem and such analysis quickly becomes complex. Frequently, the problem is reduced to a two-body problem if two simplifying assumptions are appropriate (5:11,12):

1. The bodies are spherically symmetric
2. There are no external nor internal forces acting on the system other than the gravitational forces which act along the line joining the centers of the two bodies.

Vanderburgh provides an explanation of how these basics laws are used to derive the equations used in the *Space Modeler* to determine the geocentric coordinates of all bodies modeled, both man-made and natural (42:9-17). Although Vanderburgh describes the derivation of these equations, a brief overview is required here for completeness. Furthermore, Vanderburgh's discussion pertains to elliptical orbits and does not address hyperbolic orbits. Since some deep space probes such as Galileo experience hyperbolic orbits, I present an overview of hyperbolic orbits as well.

2.2.2 Orbit Determination for Elliptical Orbits. There are at least three methods of calculating the position of a body on its elliptical orbit around the Sun at any given instant in time (26:181):

- Numerical integration.
- Obtaining the body's heliocentric coordinates (longitude, latitude, radius vector) by calculating the sum of periodic terms.
- From the orbital elements.

In (42), Vanderburgh describes the orbit determination techniques used in the *Satellite Modeler* and the *Space Modeler*. In general, near earth satellites are propagated using the SGP4 model developed at the North American Aerospace Defense (NORAD) Space Surveillance Center (SSC) and other bodies are propagated by standard equations using orbital elements. Six elements are required to describe the motion of an orbiting body. Although more than six are defined, the following are often referred to as the "classical" orbital elements (5:58) (see Figure 1):

- The semi-major axis (a) - a constant defining the size of the conic orbit.
- The eccentricity (e) - a constant defining the shape of the conic orbit.
- The inclination (i) - The angle between the equatorial plane and the orbital plane.
- The longitude of the ascending node (Ω) - the angle, in the fundamental plane, between the vernal equinox and the point where the satellite crosses through the fundamental plane in a northerly direction (ascending node) measured counterclockwise when viewed from the north side of the fundamental plane.
- The argument of **periapsis** (perigee) (ω) - the angle, in the plane of the satellite's orbit, between the ascending node and the periapsis point, measured in the direction of the satellite's motion.
- The time of periapsis passage (T) - the time when the satellite was at periapsis.

An additional element is the longitude of periapsis The longitude of periapsis is sometimes used instead of the argument of periapsis.

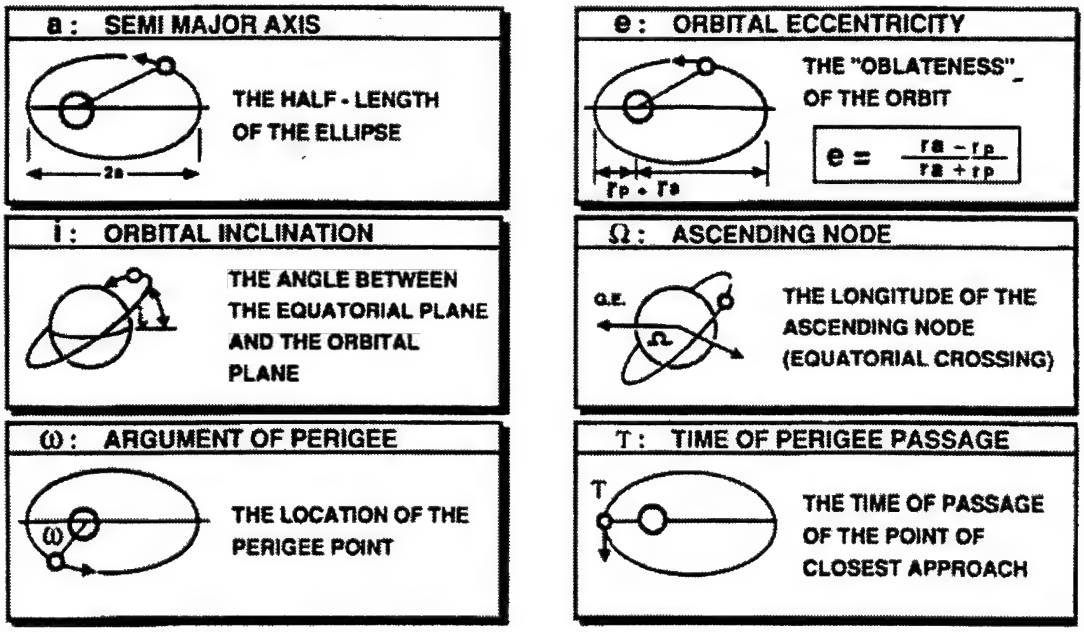


Figure 1. Orbital Elements
(23:143)

Furthermore, the **true anomaly** at epoch, v_0, may be used to substitute for the time of periapsis and could be used to locate a body at some time t_0 (5:60). In fact, the *Space Modeler* uses this approach to compute the orbits of planets, moons, asteroids, and comets.

Before proceeding, we must first establish several intermediate quantities that are based on the orbital elements and then investigate the nature of elliptic orbits. The following equations are solved once the orbital elements are known (26:214):

$$F = \cos \Omega \tag{3}$$

$$G = \sin \Omega \cos \varepsilon \tag{4}$$

$$H = \sin \Omega \sin \varepsilon \tag{5}$$

$$P = - \sin \Omega \cos i \tag{6}$$

$$Q = \cos \Omega \cos i \cos \varepsilon - \sin i \sin \varepsilon \tag{7}$$

$$R = \cos \Omega \cos i \sin \varepsilon + \sin i \cos \varepsilon \tag{8}$$

Here, ε is the **obliquity of the ecliptic**. These values are then used to determine the quantities a, b, and c and the angles A, B, and C. The following equations are used to calculate a, b, c, A, B, and C (26:215):

$$\tan A = \frac{F}{P} \tag{9}$$

$$\tan B = \frac{G}{Q} \tag{10}$$

$$\tan C = \frac{H}{R} \tag{11}$$

$$a = \sqrt{F^2 + P^2} \tag{12}$$

$$b = \sqrt{G^2 + Q^2} \tag{13}$$

$$c = \sqrt{H^2 + R^2} \tag{14}$$

To use the true anomaly in orbit determination, it must first be calculated. One method of calculating the true anomaly is to solve Kepler's equation. However, additional background is required to understand Kepler's equation. The following description of a solar (elliptic) orbit, taken from Meeus (26:182, 213-215), provides the necessary background information to understand Kepler's equation. When combined with the preceding intermediate quantities, the solution to Kepler's equation provides all the information required to determine the heliocentric rectangular equatorial coordinates of the orbiting body.

In Figure 2, the arc PKA represents one half of an elliptical orbit. The Sun, S, is located at one focus whereas the other focus, F, is an empty focus. The point C is located halfway between the **perihelion** P and the **aphelion** A.

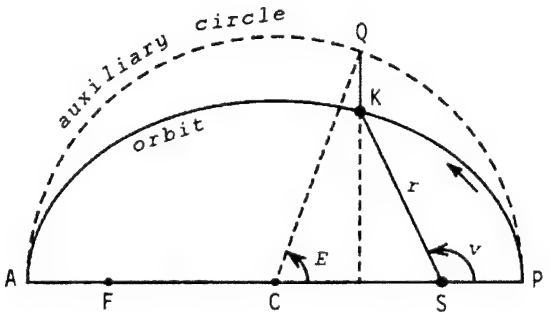


Figure 2. True and Eccentric Anomalies of an Elliptical Orbit (26:181)

Suppose that, at a given instant, the orbiting body is at K. The radius vector, r, is the distance of the body from the Sun at that instant and is expressed in astronomical units (AU). The true anomaly (v), at the same instant, is the angle over which the object has moved since the

previous passage through perihelion P. When r and v are known in reference to a given epoch, K may be determined.

The nature of an elliptical orbit is such that the body does not orbit with constant angular velocity. As a result, v does not change linearly with time. Therefore, we introduce **mean anomaly**, M, which represents the position of the body if it orbited the Sun at a constant angular velocity in a circular orbit. In other words, the mean anomaly is the angular distance the body would have moved from perihelion if it orbited the Sun at a constant velocity in a circular orbit. In Figure 3, P'SK' defines the mean anomaly at the same instant PSK defines the true anomaly. Since M changes uniformly with time, its value is easy to determine given the time of perihelion. The problem is then reduced to determining v when M and the eccentricity (e) are known.

We have particular interest in the eccentricity, the ratio CS/CP (see Figure 2), since it describes the shape of the conic orbit. When $e < 1$, as in this case, the conic is an ellipse. However, when $e > 1$, the conic is a hyperbola and the equations used to determine the orbit are different. We shall discuss hyperbolic orbits in Section 2.2.3.

To establish the connection between M and v, it is necessary to introduce **eccentric anomaly**, E. The eccentric anomaly is geometrically defined in Figure 2 by angle PCQ. Once E is known, v may be obtained from the equation:

$$\tan\frac{v}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} \tag{15}$$

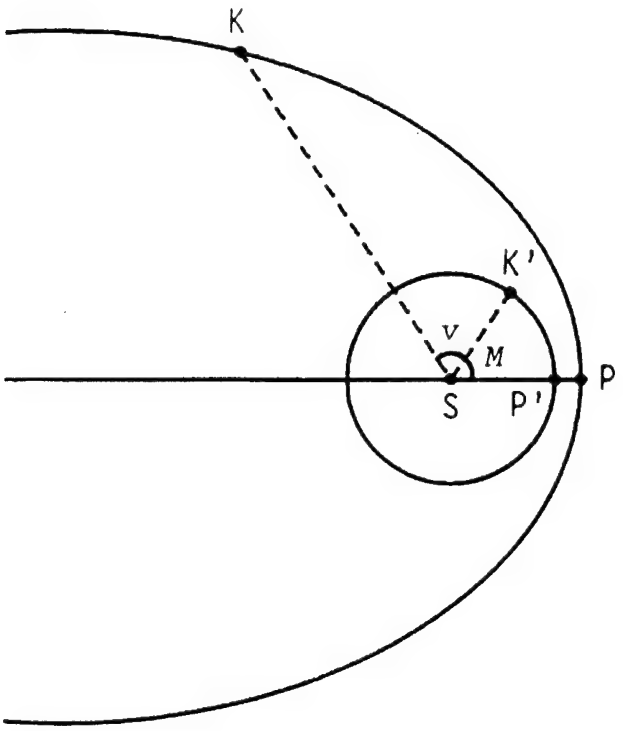


Figure 3. True and Mean Anomalies of an Elliptical Orbit
(26:183)

However, we must solve Kepler's equation to find E:

$$E = M + e \sin E \tag{16}$$

This is a transcendental equation and cannot be solved directly. Fortunately, an iterative approach can be used to compute an arbitrarily close approximation to E. This approach is called Newton iteration and is described in (26:184) and (5:221). Once the eccentric anomaly has been computed, the radius vector can also be determined from the following equation:

$$r = a (1 - e \cos E) \tag{17}$$

Finally, the heliocentric rectangular equatorial coordinates may be computed from the following equations (26:215):

$$x = r\,a \sin (A + \omega + v) \tag{18}$$

$$y = r\,b \sin (B + \omega + v) \tag{19}$$

$$z = r\,c \sin (C + \omega + v) \tag{20}$$

While Equations 18-20 are used to compute the coordinates at the time of epoch, it is necessary to determine the coordinates for an arbitrary time as well. This may be accomplished by finding the mean anomaly for the specified time and using the result in Kepler's equation (Equation 16). The coordinates are then found as previously described. Two additional equations are required. The first equation describes the daily motion of the satellite and the second is used to compute the mean anomaly at the specified time.

$$n = \sqrt{\frac{k^2 M_{Body}}{a^3 M_{Sun}}} \tag{21}$$

where

n is the daily mean motion
k is the Gaussian gravitational constant (34:696)
a is the semi-major axis of the orbit
M_{Body} is the mass of the body the satellite is orbiting (e.g., Jupiter)
M_{Sun} is the mass of the Sun

M_{Body} and M_{Sun} are frequently replaced by the ratio of the masses because astronomers can determine the ratio with much greater accuracy than the independent masses.

$$M = n(t - T_0) + M_0 \tag{22}$$

where

M is the mean anomaly
n is the daily mean motion
t is the specified time
T_0 is the time of epoch
M_0 is the mean anomaly at epoch

2.2.3 Orbit Determination for Hyperbolic Orbits. Although the orbits of most heavenly bodies such as planets and moons are elliptical, hyperbolic orbits are used to describe trajectories for interplanetary space travel. Planetary probes must travel hyperbolic paths relative to the Earth in order to escape its gravitational influence (5:38). Of course, the same principle would apply as the probe approached or departed other planets as well. As we expand our interest to include deep space satellites, it becomes necessary to compute hyperbolic orbits as well as elliptical orbits.

In Figure 4, F is the left hand focus of the hyperbola. Assume it is the location of the gravitational body influencing the orbit of a satellite. The left hand branch of the hyperbola shows the path of an object being attracted to the body whereas the right hand branch of the hyperbola shows the path of an object being repulsed by the body. The turning angle, δ, may be calculated from $\sin\frac{\delta}{2} = \frac{a}{c}$. However, since $e = \frac{c}{a}$, this equation becomes (5:39):

$$\sin\frac{\delta}{2} = \frac{1}{e} \qquad \text{where } e \text{ is the eccentricity} \tag{23}$$

Just as with elliptical orbits, the location of a body at some time, t_0, may be calculated from the anomaly. However, the daily motion for a hyperbolic orbit is computed with a different equation than in the elliptic case.

$$n = \sqrt{\frac{k^2 M_{Body}}{(-a)^3 M_{Sun}}} \tag{24}$$

where

n is the daily motion
k is the Gaussian gravitational constant (34:696)
a is the semi-major axis of the orbit
M_{Body} is the mass of the body being orbited (e.g., Jupiter)
M_{Sun} is the mass of the Sun

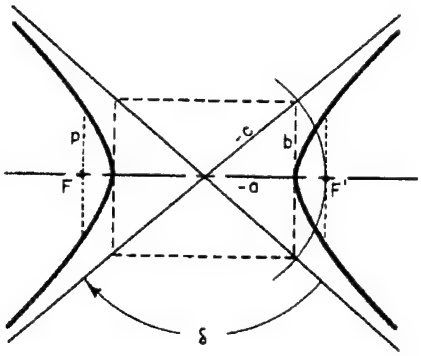


Figure 4. Geometry of the Hyperbola
(5:38)

> M_{Body} and M_{Sun} are frequently replaced by the ratio of the masses because astronomers can determine the ratio with much greater accuracy than the independent masses.

Although an understanding of hyperbolic orbits is essential to the discussion of deep space probes, the next section examines a system of satellites with elliptical orbits, the Global Positioning System.

2.3 Global Positioning System (GPS)

The Global Positioning System has revolutionized navigation whether it be air, land, or sea. This section presents a brief history of GPS, the components which comprise it, major characteristics of the transmitted signal, and describes the types of error encountered in its use as well as typical error budgets. The first topic is a brief history of the system.

2.3.1 History of GPS. Methods of navigation have been evolving for centuries. From the advent of celestial navigation techniques used by sailors to the most current electronic navigational aids found in modern aircraft, man has continually sought more accurate methods to

guide his travel across great distances. One of the most dramatic advances is the result of research and development efforts by the United States Navy and Air Force - the Navstar GPS (3). Early papers describing the concept for GPS can be traced back to the late 1970s. A review of the history and evolution of the Department of Defense's navigation technology program was given by Easton in 1978 (12). Early GPS testing consisted of a ten satellite constellation designed to provide the best coverage over the Yuma Proving Ground. The test was a success and tremendous effort was directed toward the design of the operational constellation. The GPS Program Management Directive specified system availability to be 0.98 with 18 or more satellites functioning within specifications (3). Early designs called for a constellation of 24 satellites but that number was initially cut to 18 due to budget constraints. Today, 25 satellites are operational and in good health.

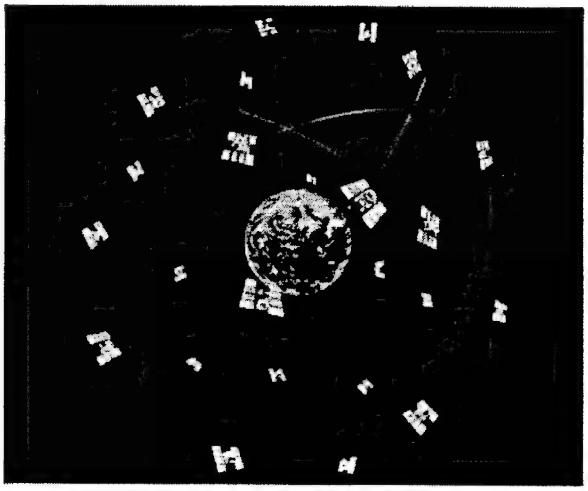


Figure 5. Navstar GPS Constellation
(23:19)

2.3.2 Components of GPS. The GPS system consists of three major segments:

- Space Segment
- User Segment

- Control Segment

The space segment currently consists of 25 satellites arranged in six orbit planes inclined 55-degrees (with respect to the equatorial plane) 10,898 nautical miles above the earth (see Figure 5). There are typically four satellites in each orbit. The number of satellites varies as some fail and replenishment satellites are launched. The constellation is designed to consist of 24 satellites, 21 for navigation and three active spares. The satellites orbit the Earth in 12 hour orbits. However, since the Earth is rotating at the same time, the satellites cover the same ground track once in approximately 24 hours, moving over the same ground location four minutes earlier each day. This constellation design typically results in seven to nine satellites being visible to a ground receiver located nearly anywhere on Earth. With the exception of special receivers used in system control, these ground receivers constitute the user segment.

The user segment consists of receivers located worldwide on the ground, in the air, and at sea. A receiver requires at least four satellites to accurately determine its position and the time. Velocity information may also be determined using the system. GPS receivers are most widely used for navigation, but special purpose receivers may be used for accurate time synchronization and frequency dissemination. As previously mentioned some receivers are part of a special set of stations used for system control in the third segment, the control segment.

The control segment, consists of a group of unmanned monitor stations located around the world and a Master Control station located at Falcon AFB, CO (see Figure 6). These monitoring stations are required because satellites do not maintain exact orbits resulting in changes to their **ephemeris constants**. The monitor stations use signals received from the satellites as input to orbital models to determine the precise location of the satellites. The monitor stations then provide this information to the Master Control station. The Master Control subsequently updates the satellites with their precise ephemeris and corrections to their on-board

atomic clocks. The satellites in turn broadcast the corrected information for use by ground receivers.

Now that we've looked at how the segments of the system work together, we'll examine the signals transmitted by the satellites.

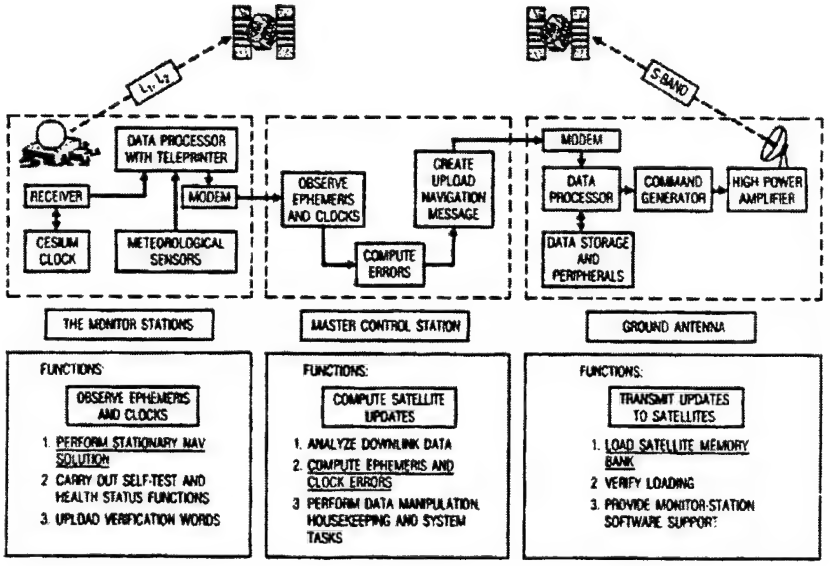


Figure 6. GPS Control
(2:4-14)

2.3.3 GPS Signal Characteristics. The GPS system is essentially two services in one. It consists of the Precise Positioning Service (PPS) and the Standard Positioning Service (SPS). The PPS requires cryptographic equipment and decoding keys. It provides predictable accuracy of 17.8 meters horizontally and 27.7 meters vertically with 100 nanosecond time accuracy. The SPS is available to the civilian community without restriction. It is intentionally degraded through the use of **selective availability** and provides predictable accuracy of 100 meters horizontally and 156 meters vertically with 167 nanosecond time accuracy.

The dual modality of the GPS system is accomplished using two distinct encodings: the C/A-code (Coarse Acquisition Code) and the P-code (Precision Code). The C/A code is available to all civilian receivers whereas the P-code is only available to a limited number of authorized users such as the US military. These signals are encoded on two separate L-band carrier waves (see Figure 7). The L1 signal carries the navigation message and the SPS signal at a carrier frequency of 1575.42 MHz. The L2 signal is exploited to estimate the ionospheric delay by PPS equipped systems and has a carrier frequency of 1227.60 MHz. The L2 signal also serves as a backup in the event L1 is lost or jammed (28). Each satellite modulates L1 once per millisecond with the C/A code, a unique 1 MHz spread spectrum pseudo-random noise (PRN) pulse. This unique PRN is used by the receiver to identify the satellite. The P-code modulates both the L1 and L2 signals and is much longer, repeating only once every seven days.

The navigation signal, a 50 Hz data stream, is superimposed on the C/A- and P-code pulse trains. This signal contains a wealth of information including the clock correction, precise ephemeris for the broadcasting satellite, and almanac information for all GPS satellites. The navigation signal, which repeats every 30 seconds, is broken down into five subframes. Each subframe lasts six seconds. Subframe 1 contains the clock correction information. Subframes two and three contain the latest values of the satellite's ephemeris. Subframes two and three also contain a figure of merit used in determining the accuracy of the user range data by the receiver. Subframe four contains navigation messages and satellite health status. Subframe five contains the almanac information for one satellite of the constellation. The almanac information is for a different satellite each broadcast resulting in an approximate 12 minute cycle time to broadcast almanac information for the entire constellation (23:25,26). All GPS position information is defined using the World Geodetic System 1984 (WGS84) reference coordinates.

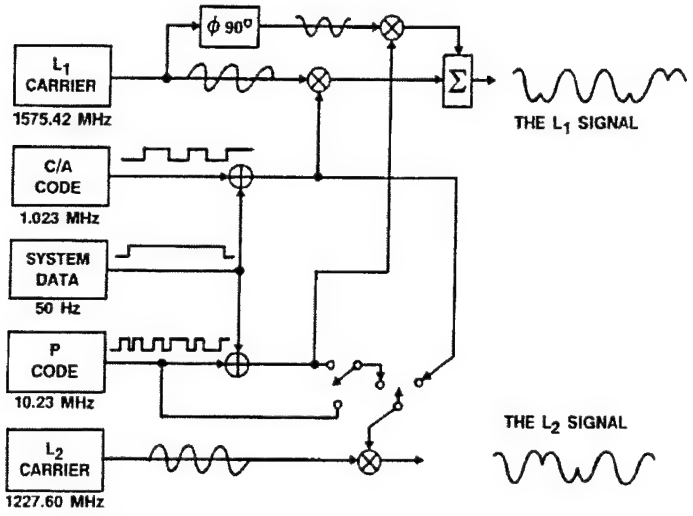


Figure 7. GPS Satellite Signals
(28:5-5)

2.3.4 GPS Error. The navigation accuracy of GPS is a function of two factors: the user range error (URE) and the system geometry. The URE is an error vector along the line of sight between a GPS satellite and a receiver. The error amplification due to system geometry is described as the geometrical dilution of precision (GDOP). When expressed in terms of three-dimensional radial error, the GPS navigation estimation (**user navigation error**, UNE) is computed by multiplying the Position Dilution of Precision (PDOP) by the URE (2:4-5) (23:59). These error components are now discussed in greater detail.

The URE encompasses errors due to the inability to accurately model space parameters that affect the range measurement, errors due to insufficient estimates of the spacecraft ephemeris and clock parameters, and line of sight modeling errors (2:4-2). It is by nature different for every single range measurement. To establish system accuracy, it is necessary to describe this error in statistical terms that are valid for every measurement. Table 1 shows the GPS URE budget for various components of the system for normal operation. Next, we'll look at the other half of the equation, the geometric dilution of precision.

Source of Error and Responsibility	*Error Sources*	*Error Quantities in Meters (1 σ)*
Space	Clock and Navigation Subsystem Stability	3.3
	L-Band Phase Uncertainty	0.5
	Predictability of SV Perturbations	1.0
	Other	0.5
	Maximum Total Segment URE	3.5
Control	Ephemeris Prediction and Model Implementation	4.2
	Other	0.9
	Maximum total Segment URE	4.3
Navigation User	Ionospheric Delay Compensation	2.3
	Tropospheric Delay Compensation	2.0
	Receiver Noise and Resolution	1.5
	Multipath	1.2
	Other	0.5
	Maximum Total Segment URE	3.6
System	**Total System URE**	**6.66**

Table 1. GPS User Range Error Budget
(2:4-3)

The geometry resulting from the positions of the visible satellites in relation to the receiver has a significant impact on the navigation accuracy. Favorable geometry occurs whenever the satellites are widely dispersed with large angles separating them (from the receiver's perspective). On the other hand, unfavorable geometry occurs whenever the satellites are in a line across the sky or when they are closely grouped. Five types of GDOP are commonly used: Geometric Dilution of Precision (GDOP), Position Dilution of Precision (PDOP), Horizontal Dilution of Precision (HDOP), Vertical Dilution of Precision (VDOP), and Time Dilution of Precision (TDOP). Table 2 defines each of these types and indicates the most

common user categories for each. As seen in the table, air and space users are most interested in PDOP. As previously mentioned, PDOP is the type of geometric error used in computing three-dimensional navigation error and is the most relevant for work associated with the *Solar System Modeler* since it is anticipated the virtual receiver will be used in aircraft. The PDOP is affected by the number of visible satellites as well as their position. The fewer visible satellites, the higher the PDOP. The highly dynamic nature of satellite navigation necessitates a statistical approach to discussing user navigation error. Figure 8 reveals seven to nine satellites are visible the vast majority of the time over a 24 hour time period. According to the GPS Standard Positioning Service Signal Specification, most of the time at least two and usually more combinations of four satellites yield a PDOP constraint of six or less for a 24 satellite constellation (16). These characteristics become important when evaluating the performance of a simulation.

Type of GDOP	*Interpretation*	*Coordinates Involved*	*Typical Interested User*
GDOP	Geometrical Dilution of Precision	Ux, Uy, Uz, Ut (3D coordinates plus time	Mostly of theoretical interest. Moving time sync users
PDOP	Position Dilution of Precision	Ux, Uy, Uz (3-D coordinates	Air-related and space-related users
HDOP	Horizontal Dilution of Precision	Ux, Uy (local horizontal coordinates)	Maritime users
VDOP	Vertical Dilution of Precision	Uz (Altitude)	Air-related users
TDOP	Time Dilution of Precision	Ut (Time)	Time sync users

Table 2. Geometrical Dilution of Precision of the Various Types (23:59)

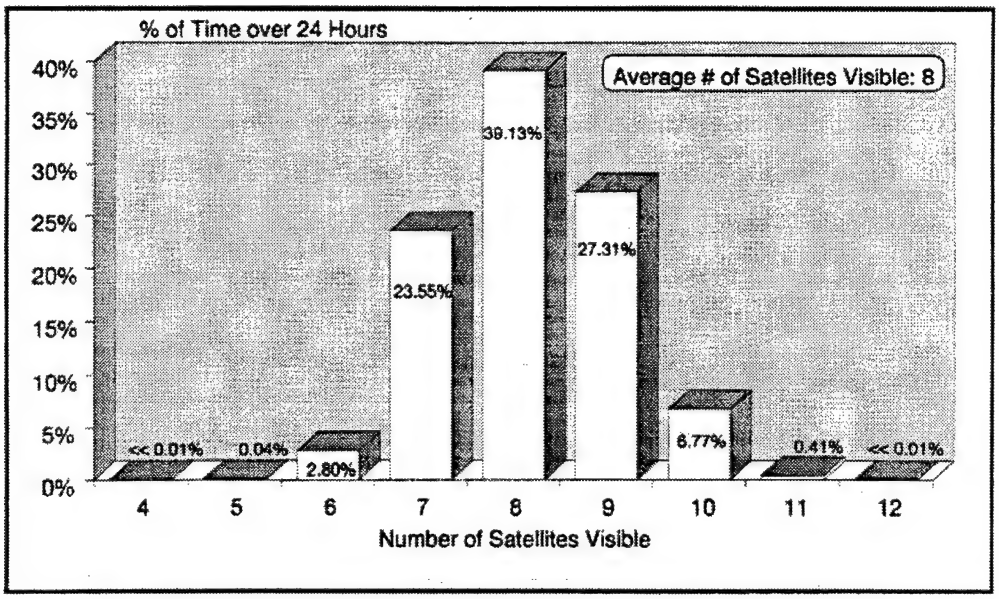


Figure 8. Satellite Global Visibility Profile
(16)

2.4 Advanced Distributed Simulation

Simulations developed in AFIT's Computer Graphics, Distributed Simulation, Virtual Environments, and 3D Medical Imaging laboratory have been compliant with Distributed Interactive Simulation (DIS) protocol standards. The *Solar System Modeler* complies with these standards as well.

The purpose of a DIS simulation is to provide for interaction with other simulations that may be running on systems located in the same room or across the country. Current research in running DIS over Asynchronous Transfer Mode (ATM) offers the potential to substantially increase the number of entities participating in a distributed virtual environment by dramatically increasing the network bandwidth (38:27). The *Solar System Modeler,* as well as other ongoing AFIT projects, are designed to participate in this environment. Participation in DIS based distributed simulations places several requirements on the participating sites: the existence of

autonomous simulation nodes, object/event based simulation, ground truth and state change information broadcast, and dead reckoning (39:148).

In (38), Stytz provides a comprehensive tutorial on distributed virtual environments (DVE). He discusses the components of DVEs to include virtual environments, software architecture, traditional simulation, networking, phenomenology, real world activity, and computer-generated actors. The reader is strongly encouraged to refer to this work for additional information. Other works providing background on distributed virtual environments include (7, 15, 17, 18, 20).

2.5 Summary

The development of a complex virtual space environment that accurately models natural and man-made entity relationships including simulated GPS navigation requires fundamental knowledge in the fields of virtual environments, astrodynamics, satellite navigation, and advanced distributed simulation. This chapter has presented an overview of these key areas.

III. General Requirements

This chapter presents the requirements for the *Solar System Modeler*. The *Satellite Modeler* laid the foundation for modeling near earth satellites such as the GPS constellation. The *Space Modeler* expanded on that work by modeling the solar system to include the Sun, planets, and moons. It also added a new user interface making the environment totally immersive. This work further extends the environment by completing the addition of comets and asteroids and adding deep space satellites, a GPS navigation capability, level of detail to planet and moon models, and a planetary tour. Background information for each of the new requirements is provided in the following sections with a summary of requirements provided in Table 3 at the end of this chapter.

3.1 Comets and Asteroids

Additional development was accomplished on the *Space Modeler* following the completion of Vanderburgh's thesis. This work included the addition of comets and asteroids. However, the implementation of the orbital mechanics algorithms was incomplete. One of the requirements for the *Solar System Modeler* was to complete this effort and accurately propagate the orbits of the comets and asteroids.

3.2 Interplanetary Satellites

A great deal of public interest was generated in December 1995 when Galileo launched a probe into Jupiter's atmosphere. Although Galileo had suffered multiple equipment malfunctions, it continued its mission of providing scientific information about Jupiter and its environment. The ability to visualize Galileo and its orbit in an immersive environment on such a historic mission

would be an obvious asset to the virtual space environment. The visualization of other deep space probes such as Voyager 2 and Pioneer 11 would further add to the rich environment of the *Space Modeler* and make the environment more complete. Consequently, these requirements were levied on the *Solar System Modeler*.

3.3 GPS Satellites and Receiver

The *Satellite Modeler* included the Earth, its moon, stars, and some satellites in near-Earth orbit. The satellites included a subset of the GPS constellation with orbital elements from 1993. While the orbital and 3D models were complete, the constellation was incomplete and the orbital elements obsolete. Furthermore, the ability to broadcast information from the satellites was limited to entity state PDUs that only provide basic information. The addition of a virtual GPS navigation capability resulted in the addition of numerous requirements for the *Solar System Modeler*. These requirements are enumerated in Table 3.

Although one requirement is to allow easy modification to accept various formats of satellite ephemerides, the initial implementation shall only accept two line element (tle) sets as provided by NORAD. This will result in greater accuracy than possible with almanac data but less accuracy than would achieved by using actual transmissions from a physical GPS receiver. This limitation was necessary to adequately scope the project. Two-line element sets and corresponding orbit determination software were available for reuse from the *Satellite Modeler*.

3.4 Planets and Moons

The planets and moons require high resolution models to provide a realistic surface when viewed in close proximity. However, when the view is moved to a greater distance, the model may retain the spherical appearance with a significantly reduced number of polygons.

The number of polygons that can be reduced while preserving an acceptable rendering of the planet or moon is proportional to the distance from the planet or moon. The *Space Modeler* did not provide varying levels of detail (LOD) for the planets and moons based on the viewing distance. This requirement was added to the *Solar System Modeler*.

3.5 Planetary Tour

Vanderburgh developed a planetary tour of the solar system following the completion of his thesis. The tour provides a sequenced visit to each of the nine planets with a short flyby of each. The tour was not a part of the *Space Modeler* and requires integration with the *Solar System Modeler*. This requirement includes modifying the user interface to provide a "start button" for the tour as well as modifying the simulation to incorporate the necessary algorithms.

3.6 User Interface

The addition of new features in the space environment requires corresponding changes to the user interface. A planetary tour option must be added to the center panel of the interface to initiate the tour. A new menu button must be added to the left panel to facilitate the selection of an interplanetary satellite as an attachment point. The trails length control on the right panel must be modified to extend trails beyond the current maximum setting of 100.

3.7 Performance

The Solar System Modeler must be capable of rendering ten frame per second when running on a four processor Onyx RealityEngine 2 system.

3.8 Summary

This chapter presented the requirements for the *Solar System Modeler*. These features, when combined with the *Space Modeler*, provide a rich environment for the exploration of the

solar system as well as virtual GPS navigation in the near-Earth environment. The next chapter provides detailed information on the system design and implementation fulfilling these requirements.

Table 3
Solar System Modeler Detailed Requirements

Requirement Area	Detailed Requirement
Interplanetary Satellites	Construct 3D graphical models of Galileo, Voyager 2, and Pioneer 11
	Load models into simulation
	Accurately propagate hyperbolic orbits
	Accurately propagate elliptic orbits
	Use level of detail (LOD) switching for models
	Display satellite locator
	Display trail of orbital path
GPS Satellites	Update constellation to consist of current healthy satellites
	Update satellite ephemeris to contain current orbital elements
	Design new DIS PDUs to broadcast DIS compliant navigation messages from satellites
	Support simulation of degraded constellation
	Ease of GPS Satellite Ephemeris Change
GPS Receiver	Receive DIS compliant navigation messages
	Decode DIS compliant navigation messages
	Accurately determine position of GPS satellites for specified time
	Identify GPS satellites visible by receiver (line of sight required for reception)
	Simulate typical error encountered by GPS receiver based on time and satellite positions
	Provide client application with GPS indicated position of receiver
	Extensible to accept multiple ephemeris types in navigation message
	Extensible to accept alternate ephemeris input interface (direct input vs. DIS)
Planets	Construct new 3D graphical models with LOD to reduce graphics rendering workload
Moons	Construct new models with LOD to reduce graphics workload
Comets	Complete accurate propagation of elliptical orbits for comets
Asteroids	Complete accurate propagation of elliptical orbits for asteroids
Planetary Tour	Integrate planetary tour modifications into Solar System Modeler
User Interface	Modify user interface to facilitate user selection of planetary tour
	Modify user interface to facilitate attachment of view to models
	Extend trail length
Performance	Frame rate of 10 frames per second or higher

IV. Design and Implementation

This chapter presents the design and implementation of the software written to meet the requirements specified in Chapter 3.

The integration of the foundational software architecture with the *Solar System Modeler* is discussed first. This includes the basic design of *ObjectSim*, the *Common Object Database* (CODB), and the *Space Modeler*. This foundation is necessary for a complete understanding of the design. Then, the design of each new or modified component of the system is presented. These components include interplanetary satellites, GPS satellites, the virtual GPS receiver (VGPSR), comets and asteroids, and the planetary tour. The virtual GPS navigation capability comprises several new classes, each of which is presented individually.

4.1 The Software Architecture

Extending the *Space Modeler* environment adds several new classes to the existing design. The `Space_Simulation` class encapsulates the entire simulation and is derived from the *ObjectSim* `Simulation` class. Figure 9 depicts the overall design of the new architecture excluding the VGPSR. The design of the VGPSR is discussed independently. The architecture of the *Solar System Modeler* is similar to the *Space Modeler* with the addition of an abstract class, `DoublePlayer`, that is derived from the `Player` class. The `DoublePlayer` class provides double precision variables for storing the Earth Centered Inertial (ECI) coordinates of entities whose coordinate values exceed single precision range. Additionally, the interplanetary satellite class is added.

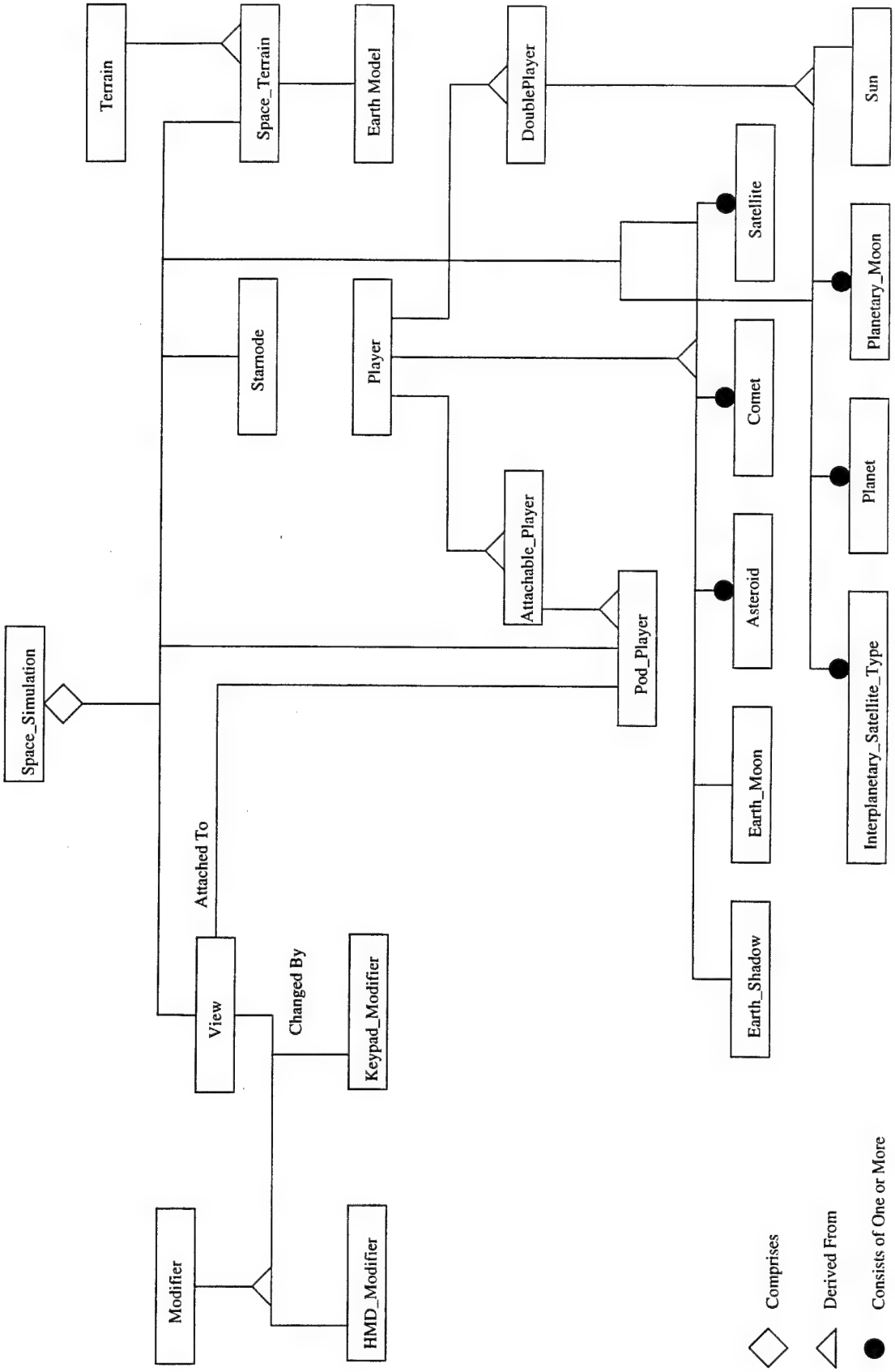


Figure 9. Object Diagram for the Solar System Modeler

Due to laboratory software upgrades, it is necessary to migrate the graphics library calls in the program from Performer 1.2 to Performer 2.0. Since the *Solar System Modeler* design is based on the *ObjectSim* framework, *ObjectSim* must be migrated to Performer 2.0 as well. Furthermore, the CODB developed earlier this year is integrated into the project for storage of data required by multiple classes and/or multiple processes. An overview of the designs of *ObjectSim*, the CODB, and the *Space Modeler* follows.

4.1.1 ObjectSim. Snyder's *ObjectSim Application Developer's Manual* provides a comprehensive description of *ObjectSim* including sample code and a class overview (37). The philosophy behind the development of *ObjectSim* was to provide a high level, reusable visual simulation framework. This framework reduces the amount of maintainable code as well as provides a proven design for the foundation of new applications. It eases the development of Performer applications by (37:A-4):

- Encapsulating and abstracting these high level simulation capabilities with C++ classes.
- Providing the basic framework and design for a visual simulation.
- Providing abstract classes which enforce interfaces between ObjectSim components.

As seen in Figure 9, the `Space_Simulation` is composed of various object classes. Most of the concrete object classes are derived from abstract classes provided by *ObjectSim*. The `Terrain` class is required for all *ObjectSim* based simulations. The terrain is based on a type of geometry file. The terrain file's local origin establishes the coordinate system origin for the entire simulation (42:22; 37). The terrain model is also responsible for establishing the lighting and the Performer earth-sky model used to present a background for the simulation. Vanderburgh discusses the implementation of the terrain in the *Space Modeler* (42). In an effort to limit extensive modifications to the *Space Modeler*, the origin of the coordinate system is not

modified in the *Solar System Modeler*. However, the addition of entities located a great distance from the coordinate system origin (Earth) revealed a problem with jitter when the viewpoint is attached to these objects. The addition of the abstract class `DoublePlayer` is an attempt to deal with this problem by adding double precision variables for storing the ECI coordinates of these entities. The problem has not been fully resolved and a possible solution is presented in Chapter 6.

Another requirement for every *ObjectSim* application is the instantiation of the `View` class. Because the view must be attached to a player that is an instantiation of the `Attachable_Player` class, in the SM, the view is attached to the user interface pod. The attachable player establishes the characteristics of the graphics window and the location of the viewpoint.

The view may be altered by a modifier instantiated from the `Modifier` class. Since the *Solar System Modeler* is an immersive application, it supports a head mounted display (HMD) as a visual interface. However, the environment may also be viewed with the system monitor. When viewed through the HMD, a head tracker is used to monitor the user's movement. On the other hand, the keyboard is used to alter the view when the monitor is used. As a result, two modifiers are required: a `HMD_Modifier` and a `Keypad_Modifier` (42).

The *ObjectSim* environment provides an excellent starting point for some types of visual simulation, such as the *Solar System Modeler*. However, some limitations are imposed by the design that do not exist when developing a simulation based directly on Performer. For a better understanding of how these limitations impact the *Solar System Modeler*, the reader is encouraged to read (37).

4.1.2 Common Object Database. The CODB is a shared memory, double buffered, semaphore protected storage container. The container may be used to store data of

predetermined types for access by multiple methods and/or processes. The class provides methods to access the database for reading and writing. Since the data is double buffered, reading and writing can take place at the same time. Using semaphores, multiple access to shared data is permitted while maintaining the integrity of the data.

The constructor for the CODB must be called once early in the application to instantiate the shared memory container. When the constructor is called, a specific structure type is specified and that structure is then "registered" with the CODB. The CODB need only be instantiated once regardless of the number of different types of data structures registered with it. However, the constructor is called each time a new structure type is registered with the CODB. As a result, multiple "new" calls may be made although only the first call actually allocates memory for the CODB.

In the *Solar System Modeler*, several CODB pointers are declared in the `global_types` header file. These pointers are used to access the CODB as various types of data are written and read. A statement such as:

```
CommonObjectDB<planet_struct> *planet_db;
```

is used to declare each pointer. A separate pointer is maintained for each type of data structure stored in the CODB. These pointers are included in the `global_types` header file to facilitate easy access to the data throughout the application. The pointers are instantiated in the constructor of `Space_Simulation`. This constructor is called once at the beginning of program execution. Statements similar to the following are used to register a data type with the CODB and assign a pointer:

```
Globals->planet_db = new CommonObjectDB <planet_struct>
(PlanetStruct);
```

CODB class methods may then be called to open the database for reading and writing.

4.1.3 Space Modeler. The *Space Modeler* is built on the *ObjectSim* framework and has a terrain, view, and numerous players. The view is modifiable by the keypad or a HMD. These concepts have already been presented in the *ObjectSim* discussion. However, there remain several aspects of the design that are important to the *Solar System Modeler* that have not been addressed. These include the coordinate system, scale, and the viewing frustrum.

The coordinates are specified in the ECI rectangular coordinate system. This coordinate system has its origin located at the center of the Earth, with the x-axis intersecting the equator at the vernal equinox, the z-axis intersecting the North Pole, and the y-axis perpendicular to the x-axis in the equatorial plane (see Figure 10) (42:27). Although the ECI system is used for orbit propagation, the DIS broadcast coordinates of all entities must be specified in the Earth Centered Earth Fixed (ECEF) coordinate system, WGS84. The WGS84 coordinate system has the z-axis parallel to the direction of the Bureau International de l'Heure (BIH) defined Conventional Terrestrial Pole (CTP) for polar motion. The x-axis is the intersection of the BIH-defined WGS84 Reference Meridian Plane and the plane of the CTP's Equator. The y-axis completes the right-handed, ECEF orthogonal coordinate system, measured in the plane of the CTP Equator, 90 degrees East of the x-axis (see Figure 11) (11:2-6). The conversion from ECI to ECEF coordinates must be made prior to DIS broadcasts.

A significant consideration in developing a virtual space environment is the scale of the model. With a solar system approximately 7 billion miles in diameter, it is important to find a scale that is large enough to adequately simulate events such as close approaches between satellites and planets, yet small enough to be processed by the computer. The *Satellite Modeler*

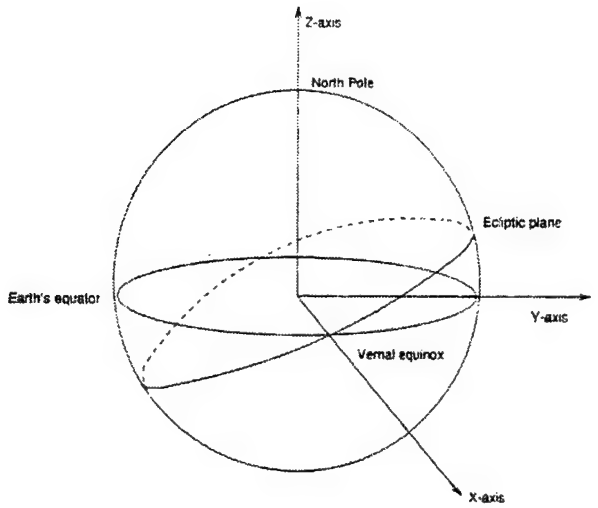


Figure 10. ECI Coordinate System
(42)

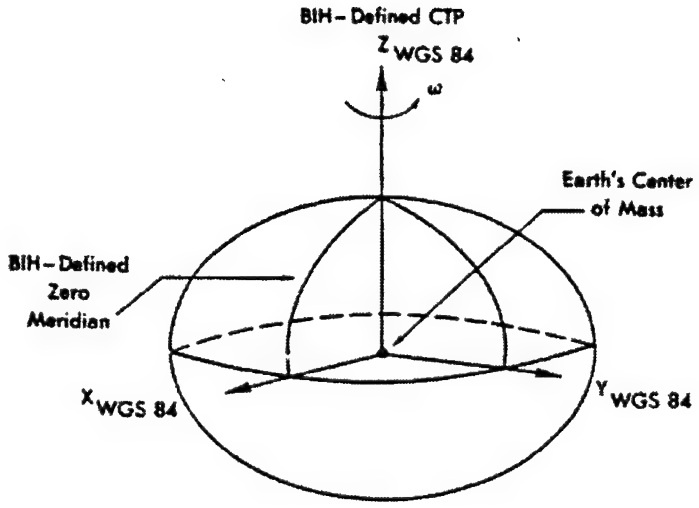


Figure 11. WGS84 Coordinate System
(11)

and *Space Modeler* used a scale of 1:10,000 and meters for units (42:29). The *Solar System Modeler* retains the same scale.

The viewing frustrum is another critical aspect of a visual simulation. Vanderburgh states the viewing frustrum far plane was set to 35,000,000 units in the *Space Modeler* because this distance coincided with diminishment of the Sun to a point light source (42:29). This setting prevents premature clipping of very large objects such as the Sun and large planets. The *Solar System Modeler* retains this setting for the far plane.

4.2 Interplanetary Satellites

The `Interplanetary_Satellite_Type` class contains the attributes and methods necessary to model the orbits of satellites that travel to other planets in our solar system. These satellites are referred to as deep space probes, deep space satellites, and interplanetary satellites interchangeably in this document. The continuous paths of these satellites are not modeled due to their complexity. Rather, orbits within the sphere of gravitational influence of a planet or the Sun are modeled. This class is derived from the `DoublePlayer` class that is derived from the *ObjectSim* `Player` class (see Figure 9).

The algorithms used to propagate deep space probe elliptical orbits are very similar to the algorithms used to propagate planetary moon orbits. However, deep space probes sometimes follow hyperbolic orbits as well as elliptical orbits. As a result, the ability to propagate hyperbolic orbits is also required. The algorithms are based on the two-body problem discussed in Chapter 2. It is assumed the perturbations of an orbit by other bodies such as moons are negligible. Although this is not always true, it is acceptable for the purposes of this simulation.

During initialization, the orbital elements are read from a unique data file for each interplanetary satellite and stored in a data structure. The `Calculate_ECI_Coordinates`

method then uses the orbital elements of the satellite to compute the ECI coordinates for a specified time. The initialization and determination of the ECI coordinates is accomplished as follows:

Initialization:

```
Read Orbital Elements from Data File

If Orbital Elements Specify True Anomaly Then

    Compute Eccentric Anomaly
```

$$E = 2a\tan\sqrt{\frac{1-e}{1+e}}\tan\frac{\upsilon}{2} \tag{25}$$

Equation 25 is Equation 15 rewritten in terms of E since we know the true anomaly and must compute the eccentric anomaly (see Section 2.2.2).

```
    Compute Mean Anomaly
```

$$M = E - e\sin E \tag{26}$$

Equation 26 is Equation 16 rewritten in terms of M since we know the eccentric anomaly and must compute the mean anomaly (see Section 2.2.2).

```
If Orbit is Elliptic Then

    Compute Elliptic Daily Motion (Equation 21)

Else If Orbit is Hyperbolic Then

    Compute Hyperbolic Daily Motion (Equation 24)
```

Calculate ECI Coordinates:

```
Determine Mean Anomaly for the Specified Time
(Equation 22)

Determine the Eccentric Anomaly for the Specified Time

    Eccentric Anomaly is found using Newton iteration
    on Kepler's equation (Equation 16) (5:221)

Solve for x and y Coordinates (34:333)
```

$$x = a(\cos E - e) \tag{27}$$

$$y = a\sqrt{1-e^2}\sin E \tag{28}$$

where

x is the x coordinate
y is the y coordinate$a$ is the semi-major axis of the orbit
E is the eccentric anomaly
e is the eccentricity

```
Rotate to ECI Coordinates (34:333)
```

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = R_1(\varepsilon)R_3(-\Omega)R_1(-i)R_3(-\omega)\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \tag{29}$$

where

$x1$, $y1$, and $z1$ are the resulting ECI coordinates
x and y are the coordinates found in Equations 27 and 28
R_i are the required rotations

```
Add the Coordinates to the Main Body Position to
     Obtain Satellite Position
```

In addition to accurately modeling satellite orbits, the simulation must present 3D graphical models of the entities being simulated. Designer's Workbench (DWB) by Coryphaeus Software was used to model Voyager II, Galileo, and Pioneer 11 (see Figures 12-14).

4.3 GPS Satellites

The GPS satellites have been part of the Lab's portrayal of outer space since the *Satellite Modeler*. They are currently instantiations of the `Sat_Player` class that is derived from the `Player` class. Since these satellites are located in close proximity to the origin of the

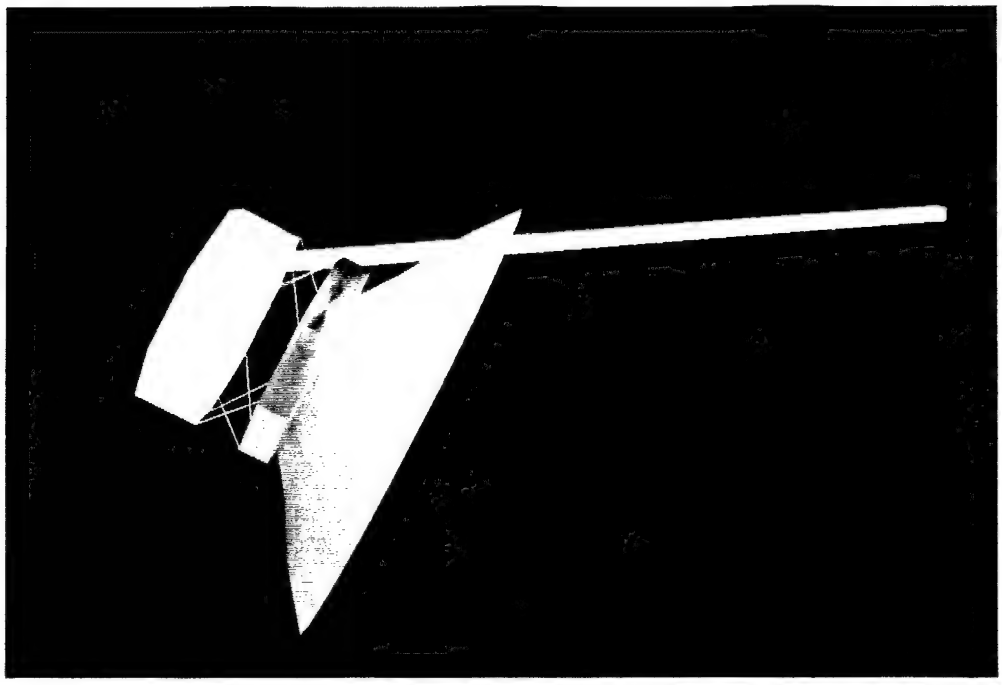


Figure 12. Model of Voyager II

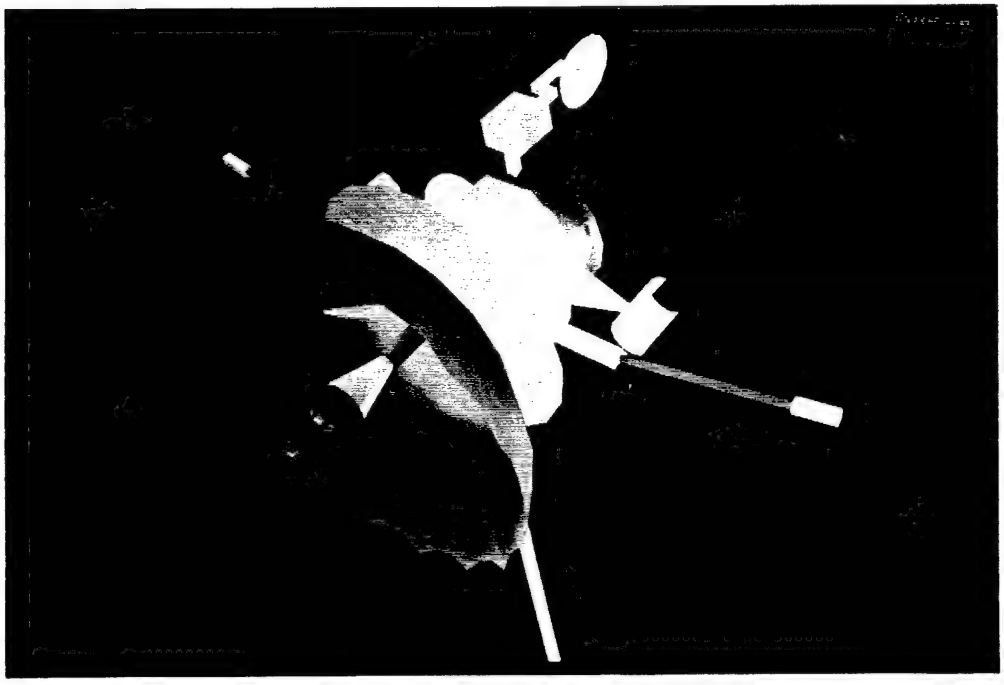


Figure 13. Model of Galileo

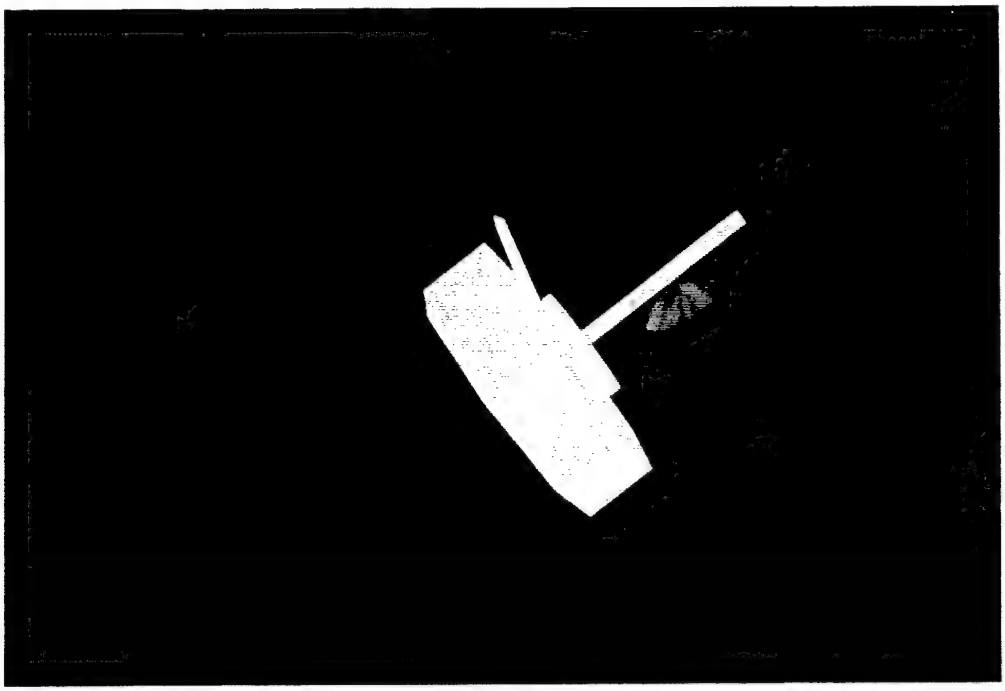


Figure 14. Model of Pioneer 11

coordinate system (the Earth), it is not necessary to modify them to be derived from the `DoublePlayer` class. However, some updates and modifications were necessary to complete the GPS constellation and place the satellites in accurate orbits.

First, the data files that specify the orbital elements of each satellite were updated to reflect the current orbital information. Additionally, the GPS constellation was updated and completed to enable virtual GPS navigation. The GPS constellation changes over time due to satellite failures and the addition of replenishment satellites. The United States Coast Guard Navigation Center website, http://www.navcen.uscg.mil/, maintains current health (status) information on the entire GPS constellation (41). The Celestial WWW website, http://www.grove.net/~tkelso/, maintains updated NORAD two-line element sets for all GPS satellites (21). Using information from these two sources, the *Solar System Modeler* contains the most current GPS orbital information available without the use of a physical receiver. This is

accomplished by updating a data file to contain only healthy satellites with their latest element sets. The program reads this data file during initialization.

The ability to broadcast Transmitter and Signal PDUs is also required for the `Sat_Player` class to support virtual GPS navigation in a DIS environment. The `Sat_Player` class includes the methods `Broadcast_GPS_Tx` and `Broadcast_GPS_Signal` for this purpose. The `Broadcast_GPS_Tx` is responsible for initiating the broadcast of the Transmitter PDUs. The Transmitter PDUs contain information such as transmission source identification, transmitter classification, modulation information, and reference system (18, 19). The Virtual GPS Receiver uses this information to identify which of the healthy GPS satellites are broadcasting. The `Broadcast_GPS_Signal` method is responsible for initiating the Signal PDU broadcasts. The Signal PDUs contain information such as transmission source identification and data (navigation message) (18, 19). As implemented in the *Solar System Modeler*, the navigation message is composed of a text string containing the two-line element set ephemeris for the transmitting satellite. The `Broadcast_GPS_Tx` and `Broadcast_GPS_Signal` methods invoke *Satellite Object Manager* routines to broadcast the necessary PDUs.

4.4 Virtual GPS Receiver

The Virtual GPS Receiver comprises six component classes and nine libraries that collectively function to provide the client application a GPS indicated position when provided with a receiver's true position and a Julian date. These classes include the `CODB_INTERFACE`, `SV_PROPAGATOR`, `SV_VISIBILITY`, `SV_SELECTOR`, `GPS_POSITION`, and `VGPS RECEIVER` (see Figure 15). The receiver functions in one of two modes: turbo or normal.

In turbo mode, a default PDOP value (refer to Section 2.3.4) of 2.4 is used for determining the standard deviation used in random variate (RV) generation (see Section 4.4.5). Three RVs are generated, one for each of the three spatial dimensions: x, y, and z. The randomly generated values are the components of a spherical model of the error encountered by an actual

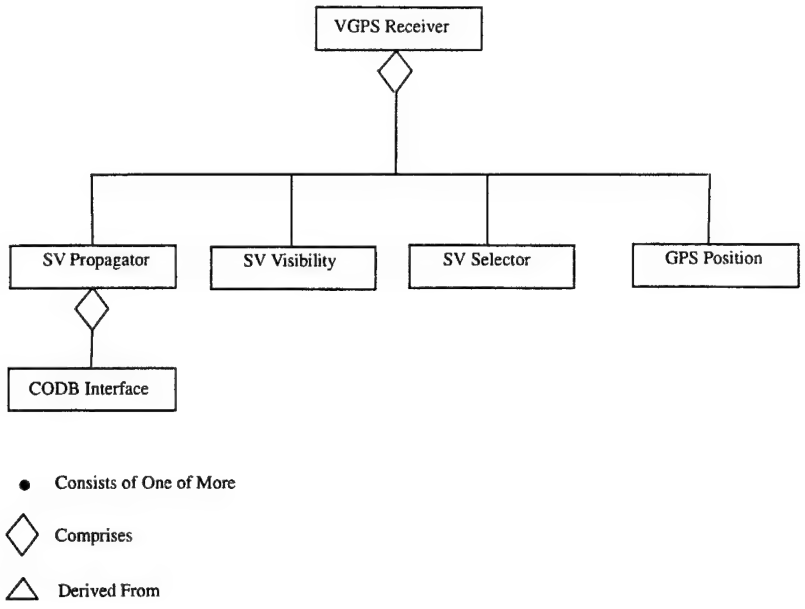


Figure 15. Object Diagram of Virtual GPS Receiver

GPS receiver. These error components are added to the x, y, and z components of the receiver's true position and the result is returned to the client application as the GPS indicated position. In this mode, no DIS PDUs are required because the entire satellite propagation cycle is eliminated to reduce compute requirements. This method generates a random GPS error based on a typical PDOP value of 2.4 rather than being based on simulated receiver-satellite geometry.

On the other hand, in normal mode, the entire GPS is simulated and PDOP is computed from the receiver-satellite geometry. This process is initiated in the *Solar System Modeler* with

the broadcast of Transmission and Signal PDUs by each GPS satellite. These broadcasts are received by the VGPSR and decoded. The receiver-satellite geometry is then determined by propagating all healthy GPS satellites, determining which satellites are visible by the receiver, computing the PDOP for all possible combinations of four visible satellites, and finally selecting the lowest resulting PDOP value. This PDOP value is then used in RV generation as in turbo mode.

In Sections 4.4.1 through 4.4.6, each component class is presented with a detailed discussion of its design and implementation. Unless stated otherwise, methods referenced in these discussions are members of the class being discussed.

4.4.1 VGPS Receiver. As illustrated in Figure 15, the VGPS_RECEIVER class encapsulates the entire receiver. It is the client application's interface to the VGPSR. The public method `get_gps_pos` is called directly by the client application when a GPS indicated position is required. The method's input parameters include the true position of the receiver, the time that the position is required, and the desired mode of operation (turbo or normal). The single output parameter is the GPS indicated position.

The algorithm in Figure 16 is implemented in `get_gps_pos` to generate a simulated GPS indicated position based on the receiver's true position. The implementation of each step of the algorithm in Figure 16 is discussed in the corresponding class presentation (Sections 4.4.3 - 4.4.6).

4.4.2 CODB Interface. The purpose of the `CODB_INTERFACE` class is to interface the VGPSR with the CODB. This allows easy modification of the program for use with alternative methods of obtaining satellite ephemeris. The `get_CODB_ephemeris` method calls an

```
If mode is normal
     Determine the position of all satellites
     Determine which  satellites are visible to the
          receiver
     Determine the lowest PDOP from the visible
          satellites
else (mode is turbo)
     Assign a typical PDOP value (i.e. PDOP = 2.4)
Determine GPS indicated position
```

Figure 16. Algorithm to Determine GPS Position

appropriate method based on ephemeris type (e.g., `get_tle` for two-line element ephemeris) to access the CODB and extract ephemeris data for all active satellites. The ephemeris data is placed in a `GPS_container` along with a `nav_message_id` and other necessary information. The `nav_message_id` is updated by the *Satellite Object Manager* each time a Signal PDU is broadcast. The `GPS_container` is only updated if a new broadcast has been received. The VGPSR is assured of using the current ephemeris by this procedure.

The `get_CODB_ephemeris` method is called from the `SV_PROPAGATOR` method `get_sv_positions`. The retrieved ephemeris is stored in a `GPS_container` that is returned to the calling method, `get_sv_positions`. To use an interface other than the CODB, the `get_sv_positions` method would simply need to call a different method that returns a properly populated `GPS_container`.

Although the VGPSR currently only accommodates two-line element set ephemeris, the `CODB_INTERFACE` class and the `GPS_container` are designed to be extended to accept

precise ephemeris (read directly from a physical GPS receiver) and almanac data. These extensions would require modifications in other components of the VGPSR as well, e.g., the SV_PROPAGATOR class.

4.4.3 SV Propagator. The SV_PROPAGATOR class encapsulates the methods required to propagate the orbits of all satellites stored in the GPS_container. The propagation of a satellite's orbit yields the position of the satellite for a specified time with accuracy dependent on the propagation model. Although currently limited to two-line element sets, this class is designed to facilitate extension to accommodate precise ephemeris and almanac ephemeris as well.

The get_sv_positions method is the public method invoked to determine the positions of all GPS satellites based on a specified simulation time formatted as a Julian date. The get_sv_positions method calls the CODB_INTERFACE method get_CODB_ephemeris to retrieve the most current data from the CODB. The propagate_all_sv method is then called to determine the type of ephemeris (currently limited to tle) and subsequently call the appropriate method to initiate propagation, i.e., tle_propagate.

The tle_propagate method iterates over all valid satellites in the GPS_container and executes a series of computations and method calls to determine each satellite's location. One method called is sgp4prop from the SGP4 library. This method is used to determine the orbits of all near earth satellites in the *Solar System Modeler.* The sgp4prop method returns the location of the satellite in ECI coordinates. The method ECI_to_ECEF is reused from the round Earth utilities library to convert the ECI coordinates to ECEF coordinates as required for compatibility with the DIS environment.

Once the satellite's positions are known in ECEF coordinates, the VGPSR must determine which satellites are visible by the receiver.

4.4.4 SV Visibility. The SV_VISIBILITY class encapsulates the methods required to determine the line of sight (LOS) visibility between the receiver and a satellite. The LOS determination is critical in simulating the GPS system because the receiver must have LOS with a satellite to receive its signal. The LOS algorithm in the *Solar System Modeler* uses geometry to determine if the satellite is below the receiver's horizon. The curvature of the Earth is computed in this model by using the WGS84 ellipsoid parameters of the Earth as opposed to assuming a sphere. Due to distortions that occur when the satellite signal travels through the thicker atmosphere near the horizon, a ten degree mask angle is imposed in the model (23). This further limits visibility of satellites not blocked by the Earth but that are positioned within ten degrees of the horizon. The algorithm is adapted from the visibility model used in the US Army Low Cost Competent Munition (LCCM) virtual prototype (31).

The public method called to determine visibility is `determine_sv_visibility`. This method's input parameters include the positions of all satellites and the receiver. The output parameter is a Boolean array identifying each satellite as visible or not visible.

Several steps are required to determine LOS between a satellite and receiver. These steps are presented algorithmically in Figure 17 followed by a discussion of the computations required to accomplish each step. Refer to Figure 18 for an illustration of the geometry. The scale of the figure is exaggerated to better illustrate the geometry. Determining the Earth's radius at the receiver first requires conversion of the receiver coordinates from WGS84 to geodetic coordinates (latitude, longitude, and altitude). The method `radius`, reused from LCCM, is then called to compute the radius of the Earth at the receiver's geodetic coordinates. The radius is

```
Determine the Earth's radius at the receiver (R)
Calculate the distance from the Earth's center to the
     receiver (d1)
Determine the angle beta (β)(includes mask angle)
Determine the angle alpha (α)
Determine LOS based on alpha and beta
if α <= π-β then visible else not visible
```

Figure 17. LOS Algorithm

depicted as R in Figure 18. Once the radius of the Earth has been determined, it is necessary to compute the distance from the center of the Earth to the receiver.

The receiver's position is specified in WGS84 coordinates and does not require transformation for this computation. The center of the WGS84 system is the center of the Earth. Consequently, the distance formula is applied to determine the receiver's distance from the center of the Earth:

$$d1 = \sqrt{x^2 + y^2 + z^2} \tag{30}$$

where

x, y, and z are the coordinates of the receiver

The next step is to determine the angle β from R and $d1$. Providing the receiver is above

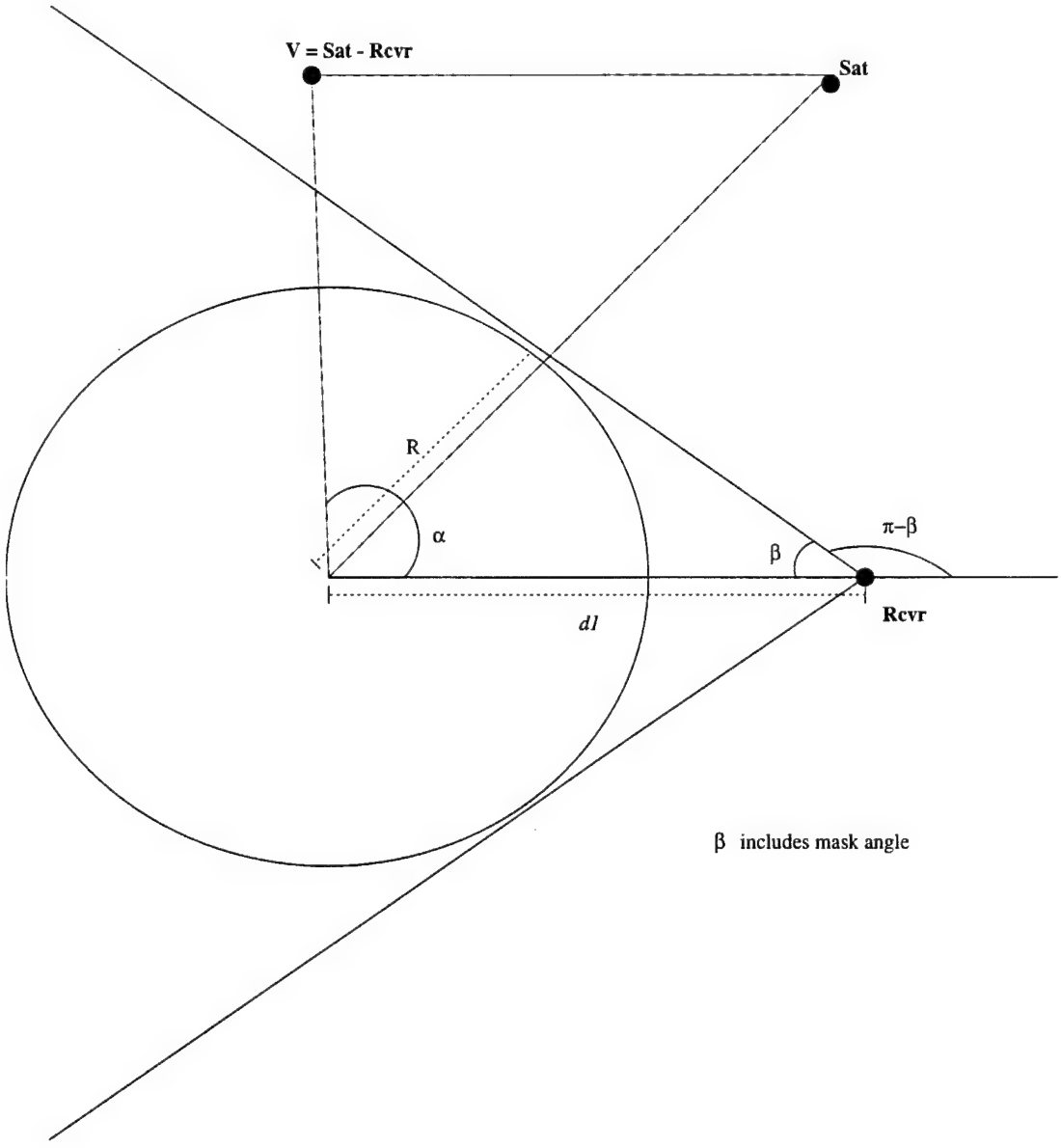


Figure 18. Satellite-Receiver Geometry for LOS Determination

the ground, *d1* will be greater than *R*. In this case, β is computed by:

$$\beta = \sin^{-1}\left(\frac{R}{d1}\right) + MA \tag{31}$$

where

β is the angle that defines the area of the receiver's view that is blocked by the Earth (including the mask angle).
R is the radius of the Earth
d1 is the distance calculated in Equation 30
MA is the mask angle

Next, we must determine the angle α corresponding to each satellite. α is measured from $\overline{Rcvr}$ to $\overline{V}$. $\overline{V}$ is determined by subtracting $\overline{Rcvr}$ from $\overline{Sat}$. Thus, the equation for computing α is:

$$\alpha = \cos^{-1}\left(\frac{\overline{Rcvr} \bullet \overline{V}}{\left|\overline{Rcvr}\right| * \left|\overline{V}\right|}\right) \tag{32}$$

Finally, LOS is determined by comparing α to π−β. If α is less than or equal to π−β, the satellite is visible, otherwise it is not. Once the visible satellites have been determined, the receiver must compute the geometric error.

4.4.5 SV Selector. The SV_SELECTOR class encapsulates the methods required to determine the lowest PDOP (using four satellites) based on the set of visible satellites found by calling the SV_VISIBILITY method `determine_sv_visibility`. The public method `select_sats` is called to find the lowest PDOP. The input parameters consist of the receiver position, an array identifying the visible satellites, and the positions of all satellites. The output parameters include the PDOP value and an array identifying the four selected satellites that generated the lowest PDOP. Selecting the satellites that generate the lowest PDOP is the last

step in determining the error based on the position of the satellites and the receiver. The remainder of the factors contributing to the GPS error are stochastic.

The procedure to determine the PDOP is reused from the LCCM project as described in (31).

The distance from each satellite (SV) to the receiver (R_i) is calculated first:

$$R_i = \sqrt{(SV_{ix} - R_{ix})^2 + (SV_{iy} - R_{iy})^2 + (SV_{iz} - R_{iz})^2} \tag{33}$$

where

SV_{ix}, SV_{iy}, and SV_{iz} are the satellite coordinates (WGS84)
R_{ix}, R_{iy}, and R_{iz} are receiver coordinates (WGS84)

Vectors are then formed for each of the four satellites using the following element definitions:

$$D_{ix} = \frac{SV_{ix} - R_{ix}}{R_i} \tag{34}$$

$$D_{iy} = \frac{SV_{iy} - R_{iy}}{R_i} \tag{35}$$

$$D_{iz} = \frac{SV_{iz} - R_{iz}}{R_i} \tag{36}$$

$$D_{it} = -1 \tag{37}$$

The vectors of the four SVs are then put in a matrix format as follows:

$$D = \begin{pmatrix} D_{0x} & D_{0y} & D_{0z} & D_{0t} \\ D_{1x} & D_{1y} & D_{1z} & D_{1t} \\ D_{2x} & D_{2y} & D_{2z} & D_{2t} \\ D_{3x} & D_{3y} & D_{3z} & D_{3t} \end{pmatrix} \tag{38}$$

The next step in determining the PDOP is to perform the transpose of matrix D. Then, the product of D and D^T are taken and the result is designated matrix E. The inverse of E is then

determined to form the matrix F. Thus, the equation for matrix F is

$$F = E^{-1} = \left[D * D^{T}\right]^{-1} \tag{39}$$

Finally, the PDOP is defined as the sum of the first three terms along the main diagonal of F.

$$PDOP = \sqrt{F_{0x} + F_{1y} + F_{2z}} \tag{40}$$

The fourth diagonal element, F_{3t}, is not part of the position error and therefore is not in the equation.

The PDOP must be found for each interrogation of the GPS receiver due to the continuous rotation of the Earth and movement of the satellites in space. The PDOP value is used by the GPS_POSITION class in the generation of the GPS indicated position as discussed in the next section.

4.4.6 GPS Position. The GPS_POSITION class encapsulates the methods required to compute a realistic random error that is added to the true position of the virtual GPS receiver. The public method `determine_gps_position` is called to accomplish this task. The method's input parameters include the receiver's true position; the PDOP as determined by the SV_SELECTOR method `select_sats`; and deltaT, the typical time between receiver interrogations. The output parameter is the GPS indicated position. This class is used regardless of whether the receiver is in turbo or normal mode. The input parameter deltaT is used in a function that correlates the change in error to the elapsed time between receiver interrogations.

As discussed in Chapter 2, the user navigation error (UNE) is expressed in terms of three-dimensional radial error. Since the receiver must return the indicated position in terms of

x, y, and z components, it is necessary to find a relationship between the radial error and its individual components. Three assumptions are necessary to statistically derive this relationship. These assumptions are typical in developing statistical relationships of navigation errors (2:4-4). It is assumed the errors along the three mutually orthogonal axes are: 1) unbiased, 2) normally distributed with equal variance, and 3) uncorrelated. Given these assumptions, the following procedure is used to derive the relationship:

1. Microsoft Excel V5.0 is used to generate 1000 uniform random numbers for each of the three axes.

2. Each uniform random number is then used as the random number input (R) to Schmeiser's approximation to the inverse cumulative distribution function for the standard normal distribution:

$$X = F^{-1}(R) \approx \frac{R^{0.135} - (1-R)^{0.135}}{0.1975} \tag{41}$$

where

accuracy is one decimal place in the range 0.0013499 <= R <= 0.9986501 (4:335).

3. The output of Schmeiser's approximation is multiplied by a specified standard deviation to approximate a normal distribution with that standard deviation.

4. Steps 1 through 3 are iteratively applied to generate 1000 x, y, z triples for sixteen different standard deviations (1..16) resulting in sixteen normal distributions.

5. The distance formula (Equation 31) is then used to compute the corresponding radial error for each of the (x, y, z) triples of each normal distribution.

6. The standard deviation of the resulting 1000 radial errors is then computed for each of the sixteen normal distributions. (See Appendix 1 for summary statistics of the randomly generated radial errors.)

7. Linear regression is used to analyze the relationship between the standard deviations of the components and the corresponding standard deviation of the radial errors for each distribution (see Figure 19). The following relationship between component standard deviation and radial error (UNE) standard deviation results:

$$\sigma_x = \sigma_y = \sigma_z \approx \frac{\sigma_{UNE}}{0.688} \tag{42}$$

This relationship is used to determine the standard deviation that is multiplied by Schmeiser's approximation to the standard normal distribution for generating the component errors.

8. Before the component errors can be added to the receiver's true position, they must be correlated to previously generated errors. The correlation filter is derived through a discrete time Fourier transform. This process assumes independent, identically distributed random variables (the error) periodically with period T (deltaT). The resulting filter requires an initial error value, y_0, with subsequent error values determined by recursively applying the equation (10):

$$y_{k+1} = e^{-.13863*deltaT} y_k + (1 - e^{-.13863*deltaT}) x_k \tag{43}$$

where

y_{k+1} is the new correlated error

y_k is the last correlated error

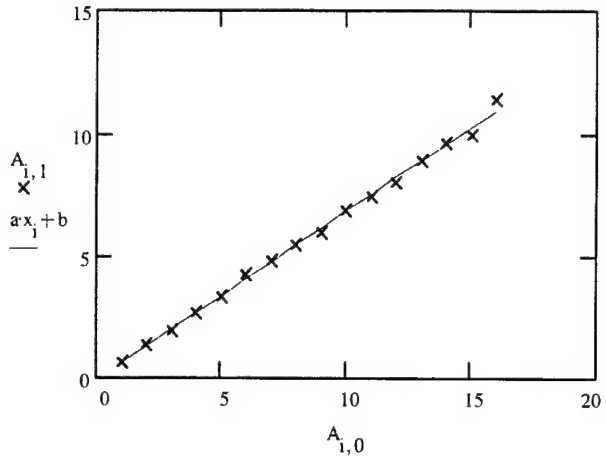
x_k is the randomly generated error

9. After the correlation filter is applied to each error component (x, y, and z), the result is added to its corresponding true position component resulting in the GPS indicated position.

As this discussion has shown, the final GPS indicated position is the true position with a randomly generated, normally distributed, time-correlated error model applied to it resulting in a realistic simulation of the error encountered in actual GPS navigation.

4.5 Summary. This chapter presented the design and implementation of the *Solar System Modeler.* It included a discussion of the underlying software architecture, deep space probes, GPS satellites, and the VGPSR. The implementation of this design resulted in a simulation capable of modeling GPS navigation and visualizing the orbits of deep space probes. The next chapter presents the overall results.

Component Std Dev	Radial Std Dev
1	0.6806
2	1.3602
3	1.9863
4	2.682
5	3.324
6	4.2036
7	4.8265
8	5.4878
9	5.9603
10	6.8659
11	7.4514
12	7.9612
13	8.8503
14	9.6096
15	9.9864
16	11.4043



a := slope(x, y) a = 0.688

b := intercept(x, y) b = −0.056

Regression Statistics	
Multiple R	0.998730762
R Square	0.997463135
Adjusted R Square	0.99728193
Standard Error	0.170920608
Observations	16

Figure 19. Linear Regression Analysis of Component Error

V. Results

Two of the most important elements of the *Solar System Modeler* are the graphical renderings of the modeled entities and the accuracy of the mathematical models used to simulate orbits. In this chapter, photographs of the simulation are used to demonstrate the quality of the rendered environment. In addition, statistical analysis and direct comparison with NASA supplied coordinates provide data useful in validating the accuracy of the simulation.

5.1 Interplanetary Satellites

Several requirements were associated with adding interplanetary satellites to the virtual space environment (see Table 3, Chapter 3). The first of these requirements was to construct 3D graphical models of the Galileo, Pioneer 11, and Voyager II satellites. These models were constructed using Designer's Workbench and were based on photographs of the satellites. The *Solar System Modeler* loads the models during the initialization sequence. Figures 20 - 22 illustrate the satellites in their respective mid-December 1996 orbits. During this time, Pioneer 11 and Voyager II are in hyperbolic orbits at the outer fringes of the solar system whereas Galileo is in an elliptical orbit near Jupiter. The solar system map in the user interface is particularly useful in visualizing the position of deep space satellites. The red triangle in Figure 23 depicts the position of Pioneer 11.

Modifications to the user interface facilitate interaction with the interplanetary satellites. One such addition, the ability to tether (attach) to each satellite required modifying the left panel to add a *DS Probes* button. When depressed, the names of the available interplanetary satellites are displayed (see Figure 24). After selecting a satellite, it is displayed in the center of the front panel view.

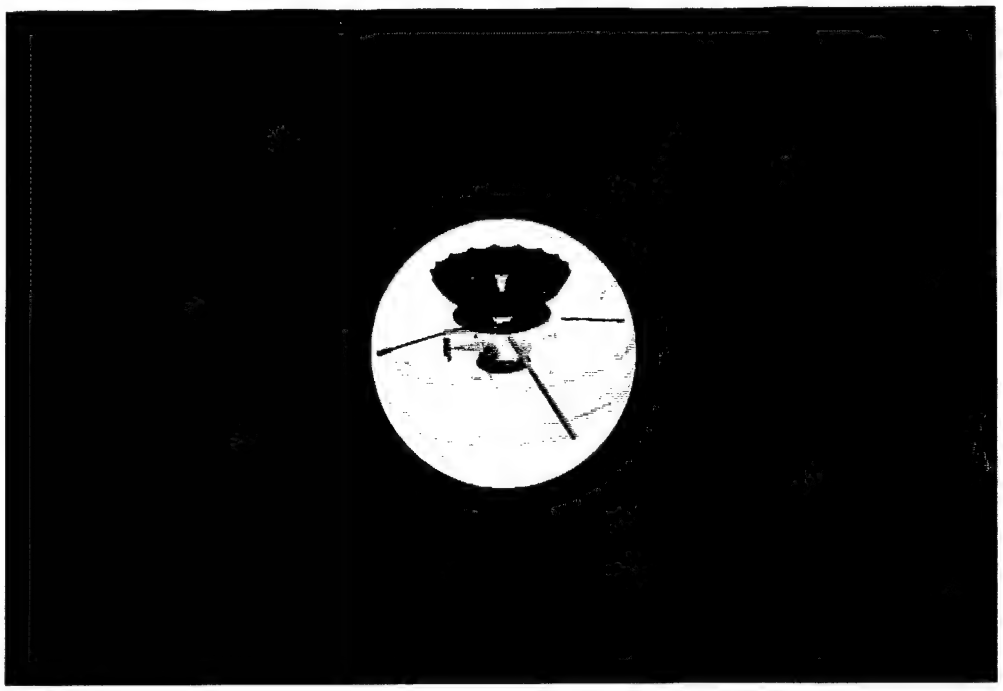


Figure 20. Deep Space Probe Galileo Orbiting Jupiter

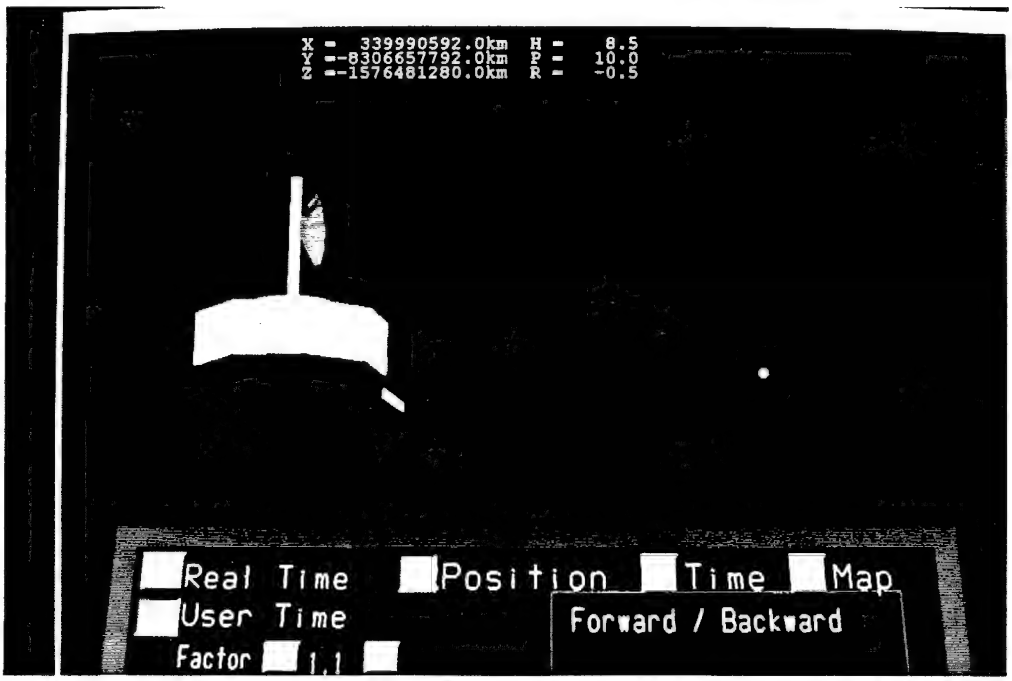


Figure 21. Pioneer 11 in Deep Space Orbit

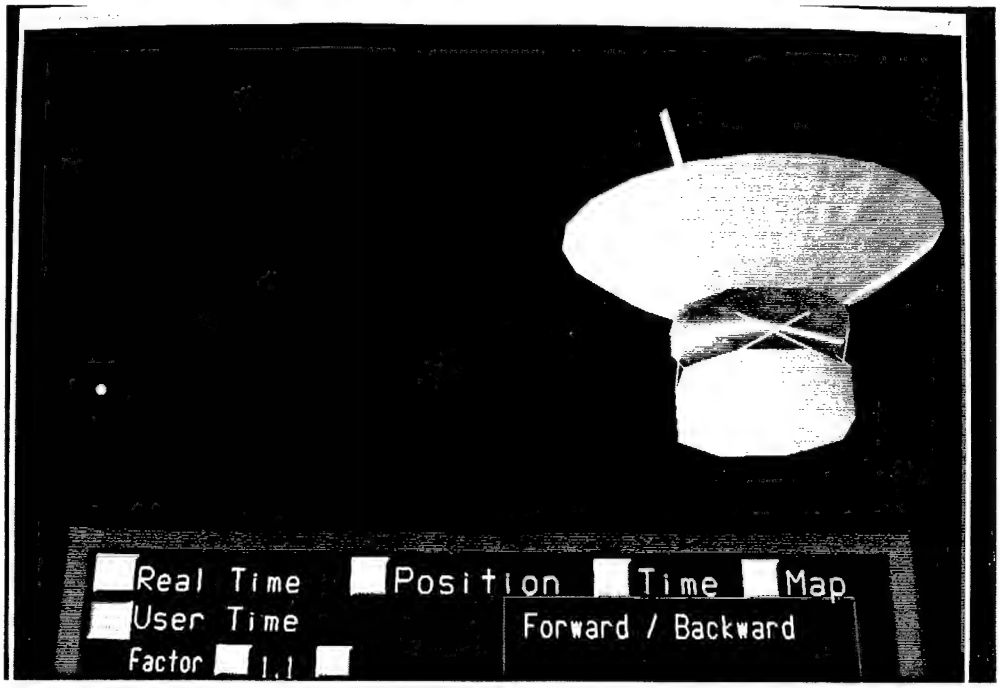


Figure 22. Voyager II Nears the Edge of the Solar System

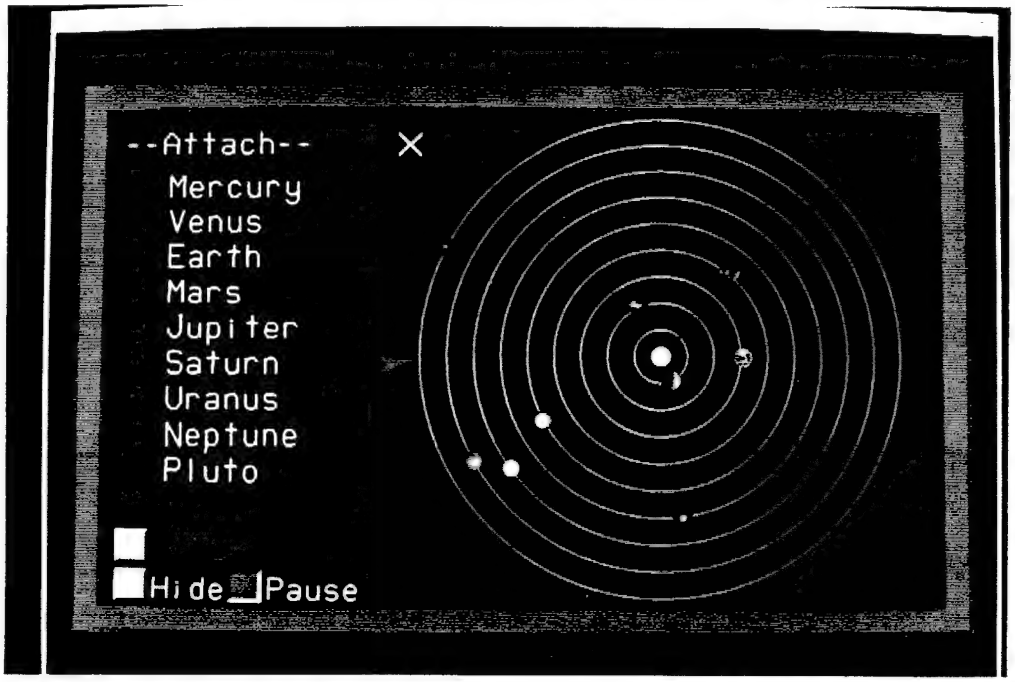


Figure 23. Locating Pioneer 11 Using the Solar System Map

Another selectable feature is the display of trails behind the satellites so their paths may be observed. Furthermore, a locator cube is used as a visual cue to assist in finding the satellites when they are at great distances. A sequence of photographs illustrating Galileo's close encounter with Jupiter's moon Europa demonstrates these features (see Figures 25 - 27).

In Figure 25, Galileo's locator is seen in front of Jupiter as the satellite encounters Europa. Figure 26 reveals Galileo is closing the distance to Europa and is now visible without its locator. Finally, in Figure 27 we see Galileo and Europa in close proximity. The trails behind Galileo and Europa during the sequence aid the user in understanding the differences in orbital motion.

In addition to providing a quality visual display, the simulation accurately models the orbits of all displayed celestial bodies. Both Vanderburgh and Kunz documented the accuracy of the orbits developed in the *Space Modeler* and *Satellite Modeler* (42, 22). However, the interplanetary satellite class is new in the *Solar System Modeler* and required validation. I contacted Mr. Lou D'Amario, NASA's Galileo Navigation Team Chief, and asked for data to assist in validating the orbit of Galileo. His team provided forecasts of Galileo's position at 00:00:00.00 Eastern Time for the 2nd, 6th, 10th, and 14th of December 1996. These positions are specified in the EME2000 coordinate system (Jupiter centered, Earth equator and equinox of J2000). As shown in Table 4, the average difference between the NASA forecasted position and the *Solar System Modeler's* propagation is 1,153 km. The average distance from the center of Jupiter to Galileo for the test cases is approximately 5,152,000 km. These values indicate the *Solar System Modeler's* accuracy is approximately 99.98%.

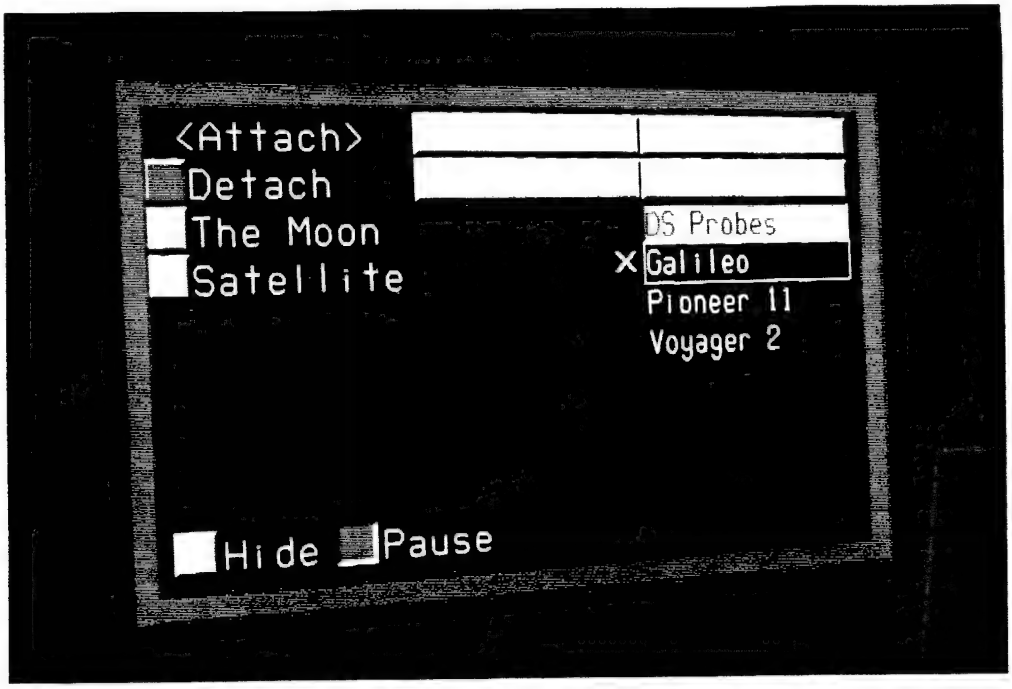


Figure 24. DS Probes Selectable from Left Panel

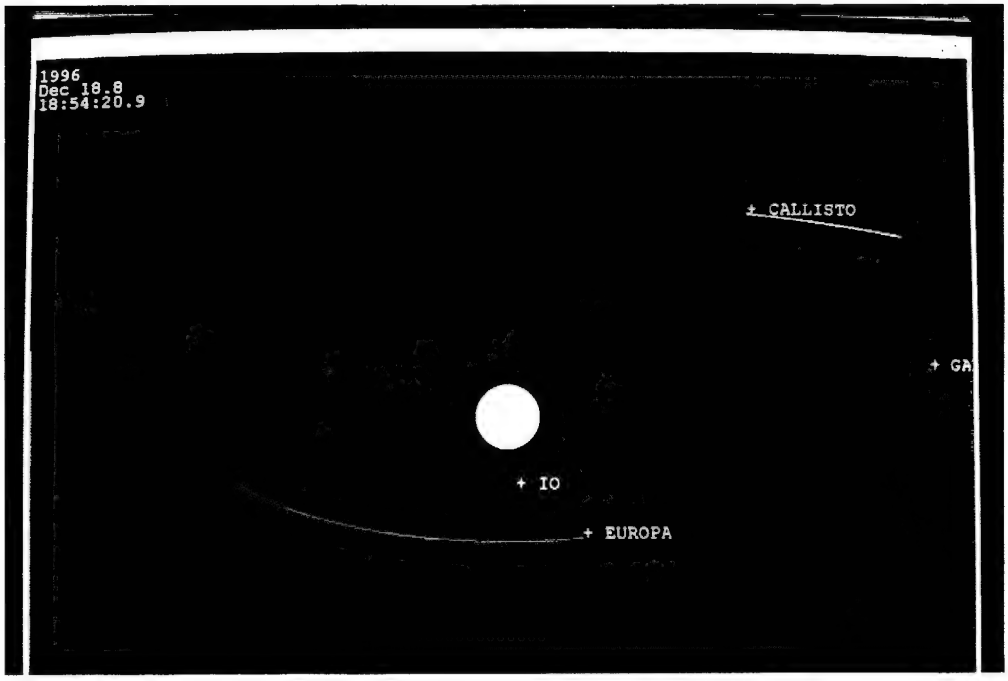


Figure 25. Galileo's Locator Visible Approaching Europa

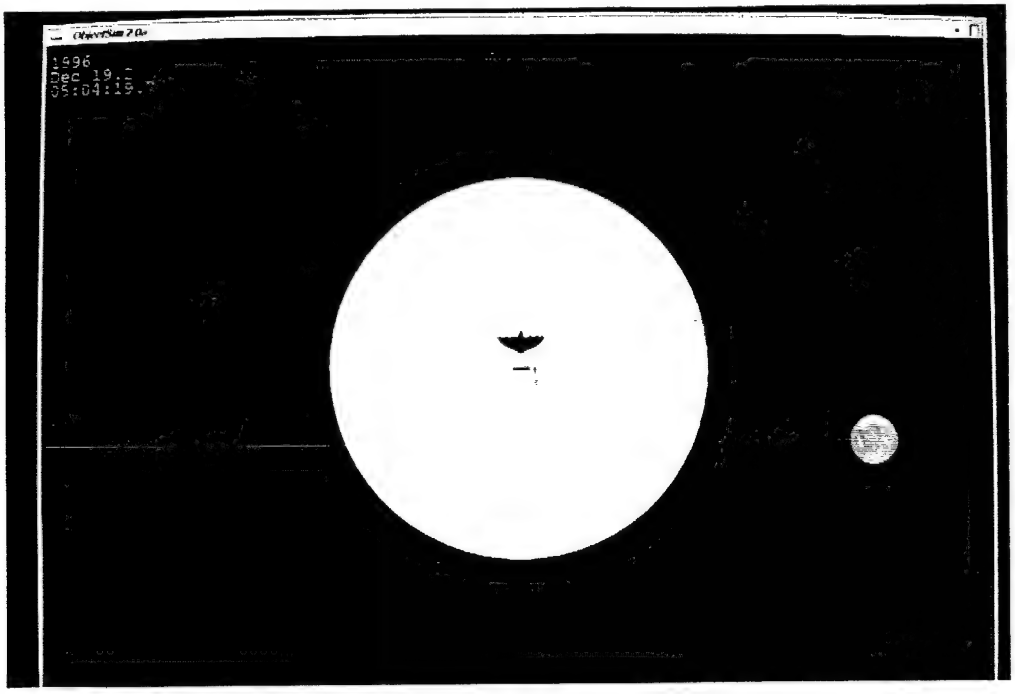


Figure 26. Galileo Nearing Europa

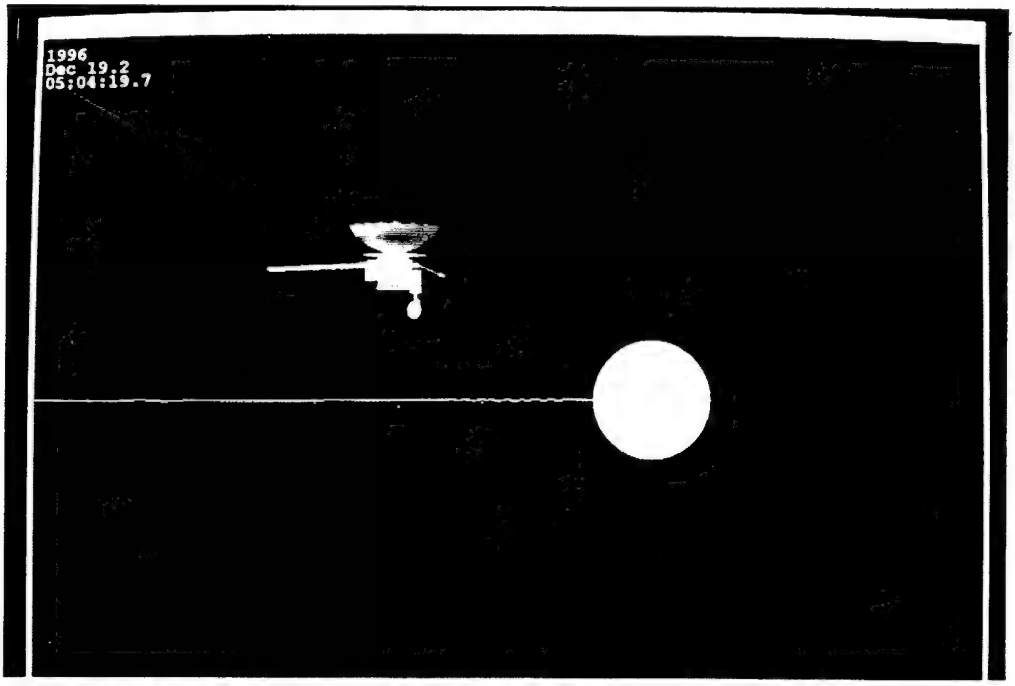


Figure 27. Galileo's Close Approach with Europa

Table 4
Comparison of NASA and Solar System Modeler Coordinates for Galileo
(Jupiter Centered, Earth Equator, Equinox of J2000)

	2-Dec-96	*6-Dec-96*	*10-Dec-96*	*14-Dec-96*
NASA Specified Position (km)				
X	6,167,722	5,725,894	4,814,568	3,249,461
Y	-698,264	-32,623	632,343	1,197,551
Z	-194,884	107,013	398,576	629,516
Solar System Modeler Calculated Position (km)				
X	6,167,312	5,725,091	4,813,790	3,248,192
Y	-698,851	-33,407	631,448	1,197,068
Z	-195,163	106,626	398,123	629,236
Difference (km) $\text{sqrt}((X_n-X_s)^2+(Y_n-Y_s)^2+(Z_n-Z_s)^2)$	768	1,187	1,269	1,386
Average Difference (km)		1,152.85		

5.2 Virtual GPS Navigation

The *Solar System Modeler* also provides the capability to navigate in the near Earth virtual environment using the Global Positioning System. Two major components provide this capability: a constellation of satellites transmitting navigation messages and a receiver to receive and decode the messages.

5.2.1 GPS Satellites. Four separate test simulations were run to verify the function of the virtual GPS system. In three of these tests, all 25 active GPS satellites in the *Solar System Modeler* broadcasted navigation messages using DIS protocols (see Figure 28). In the fourth test, the GPS constellation was degraded so only 19 satellites were broadcasting. The fourth test served only to verify the ability of the GPS system to function with a degraded constellation. It was not used to collect data for analysis. The satellites were broadcasting updated ephemeris as discussed in Chapter 4.

The first two tests were conducted using VGPSR test programs. The test programs read two data files containing receiver coordinates and time intervals respectively. The data was recorded during a ModSAF simulated F16D flight over Fort Knox, KY. The ModSAF generated flight provided 92 coordinate/time data points. For the flight, the F16D was configured to fly at 30,000 feet with a constant velocity of 0.95 mach. AFIT's Virtual Cockpit (VC) was integrated with the VGPSR for the third test. Here, a GPS guided bomb drop was simulated with the receiver located in the bomb itself. In this test, 237 coordinate/time data points were recorded. In all cases, the broadcasts were successfully received by the VGPSR.

5.2.2 GPS Receiver. In addition to verifying the ability to receive navigation messages, the three full constellation tests were designed to provide data for the analysis of the error distributions generated by the receiver. This analysis required the recording of all receiver true positions and GPS indicated positions for each position determination (range measurement) during each of the tests.

The first two tests were accomplished using the test program with the F16D flight. In one case, the receiver was configured to operate in turbo mode. In the other case, it operated in normal mode. The integrated test with the VC was run with the receiver in normal mode. The difference between the true position and the indicated position (ΔP) is the amount of simulated error introduced by the receiver. The ΔPs were determined for each individual vector component (that is x, y, and z) as well as for the total vector distance. In addition, the PDOP values computed for each range measurement were recorded, except in the case where the receiver was in turbo mode. In turbo mode, the PDOP is always 2.4. The results of these tests are summarized in Table 5.

The data in Table 5 shows that the mean error was within the system specifications for the PPS service. Furthermore, a correlation between the mean error and the PDOP value is

evident. The mean error increases and decreases with the PDOP value. This result is consistent with the VGPSR design and actual GPS performance. Comprehensive summary statistics for the tests are provided in Appendix 2.

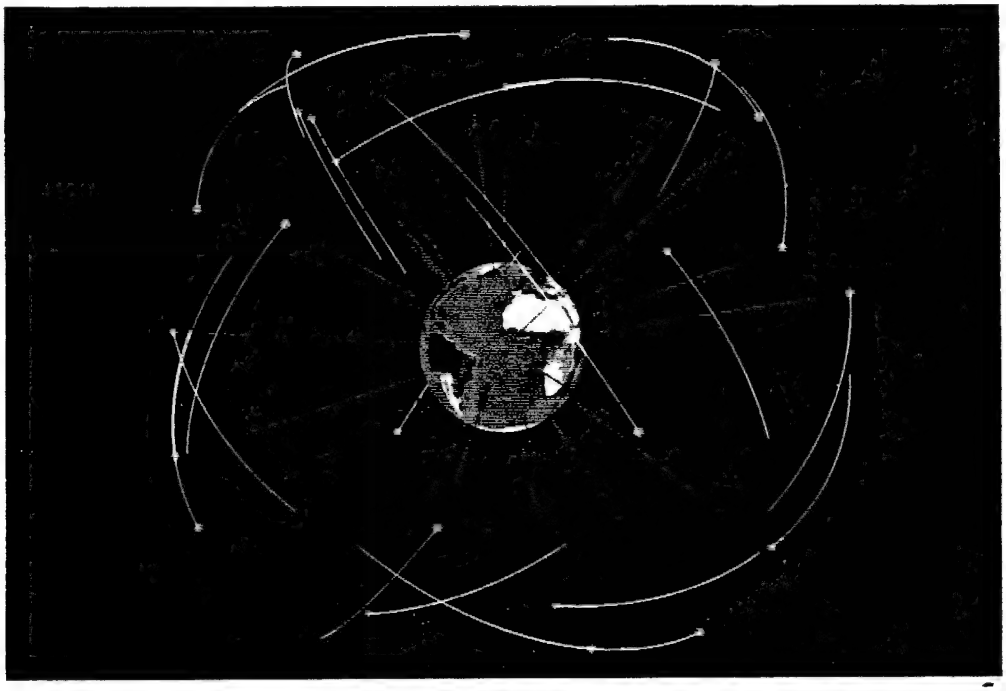


Figure 28. The GPS Satellite Constellation

Table 5
Summary Statistics for Virtual GPS System Tests

	Mean Error (m)	Standard Deviation of Error (m)	Mean PDOP
VC Bomb Drop	13.37	5.27	3.45
F16D / Rcvr Normal	9.66	4.85	2.69
F16D / Rcvr Turbo	8.59	4.25	2.4

5.3 Level of Detail (LOD) for Models

The planet and planetary moon models were reconstructed with level of detail using Designer's Workbench (DWB). Each model has three levels of detail with switching based on viewpoint distance from the model. The transition between LOD models is not detectable by the user because the switching distance was determined independently for each model through visual testing. Figures 29-32 illustrate the three levels of detail for Mars. The apparent variation in resolution is representative of all models with LOD.

5.4 Comets and Asteroids

The elliptic orbital propagation of the comets and asteroids is based on the same algorithms as the deep space probes that were discussed in Chapter 4. The accuracy of these algorithms was validated by *Solar System Modeler* propagated position comparisons with NASA forecasted positions for Galileo.

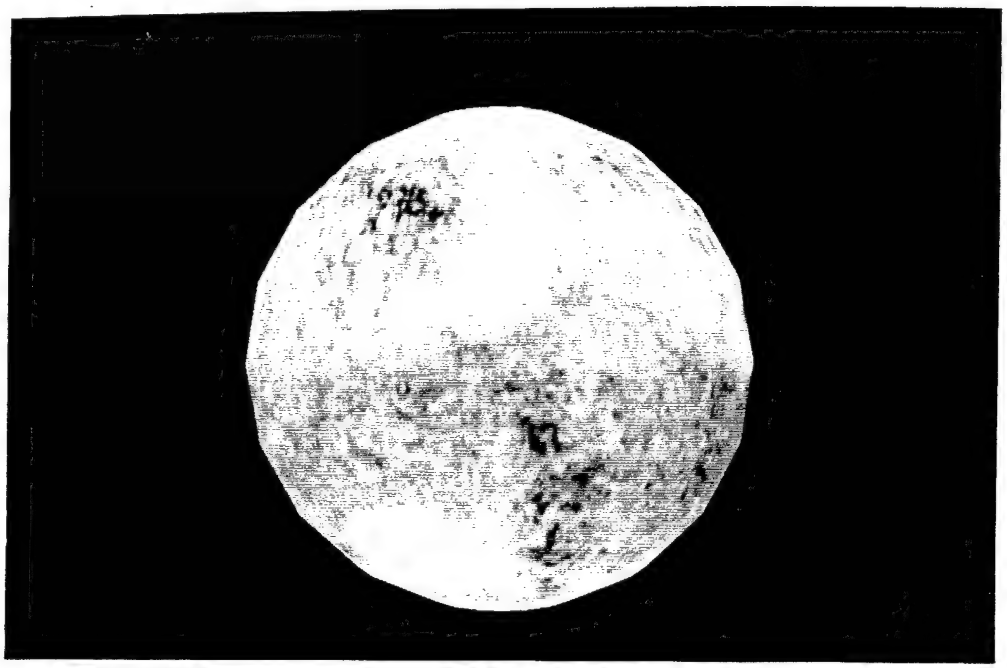


Figure 29. Lowest LOD Model of Mars

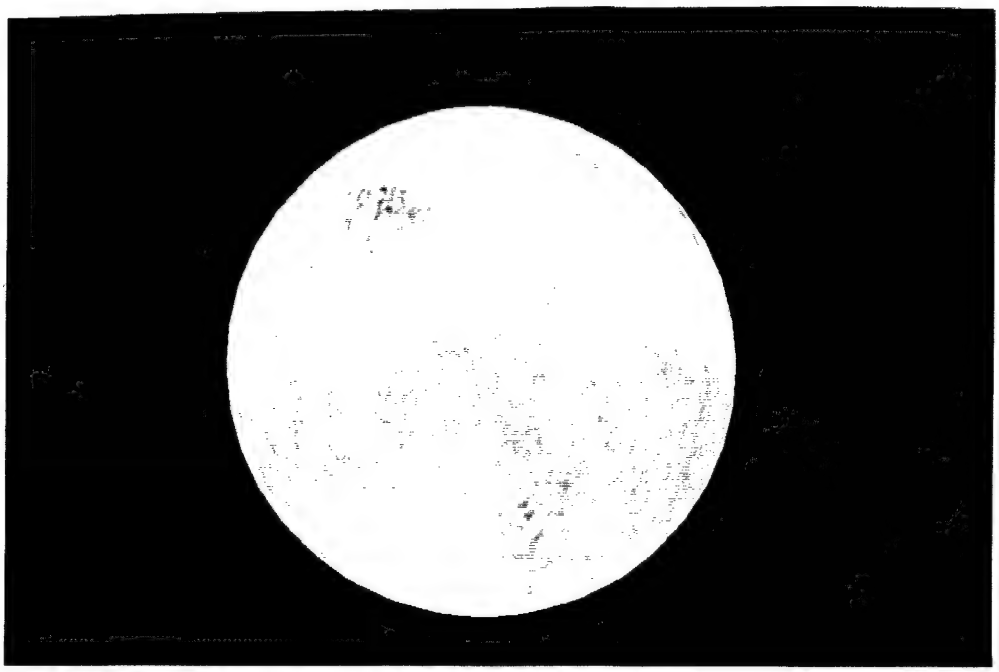


Figure 30. Intermediate LOD Model of Mars

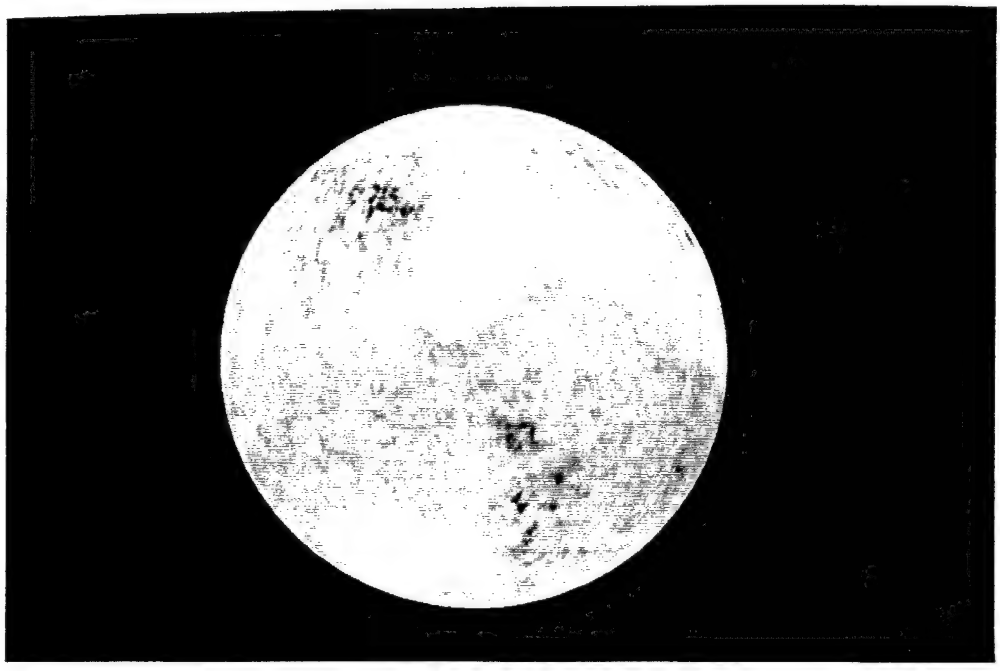


Figure 31. High LOD Model of Mars

5.5 Planetary Tour

The planetary tour parameters that adjust the initial viewpoint distance from each planet and the duration of each flyby were determined by executing the tour and subjectively adjusting each parameter for a suitable presentation. Each planet's settings are independently adjustable. The current parameter settings are based on the system performance of a four processor Onyx RealityEngine 2.

5.6 Performance

The *Solar System Modeler* frame rate varies depending on the rendered scene. Running on an SGI Onyx RealityEngine 2 system with four processors, the frame rate varies between ten and twelve frames per second.

5.6 Summary

Two important criteria in evaluating a virtual environment such as the *Solar System Modeler* are its ability to present rendered scenes that entice the user to immerse himself or herself in the environment and accurately simulate the movement of the entities displayed. In this chapter, I've included numerous photographs of the display as well as analytical results to provide the information necessary for the reader to assess the work. The next and final chapter presents conclusions and recommendations for future work.

VI. Conclusions and Recommendations

6.1 Overview

This thesis presented the *Solar System Modeler*, a distributed virtual environment for space visualization. The *Solar System Modeler* extended the *Space Modeler* by enhancing the user interface, completing the addition of asteroids and comets, adding interplanetary satellites, and significantly enhancing the GPS environment to provide virtual satellite navigation. The addition of satellite navigation added a new dimension to the project by providing useful information to other distributed environments along with the software necessary to use that information for the first time.

I presented a brief history of virtual environments along with background information on astrodynamics, the Global Positioning System and DIS in Chapter 2. In Chapter 3, the requirements for the *Solar System Modeler* were specified. Chapter 4 described the design and implementation of the software written to fulfill the requirements put forth in Chapter 3. Finally, in Chapter 5, I presented the results of the work to include analysis supporting the validity of the algorithms developed and implemented in fulfilling the requirements. Although the results indicate the requirements have been met, several areas that could be improved became apparent as I worked on the project.

6.2 Recommendations for Future Work

The *Solar System Modeler* displays a rich environment providing an excellent tool for visualizing space. It is particularly well suited for exploring the orbital relationships between entities within several million kilometers of each other. Nevertheless, several areas of the program leave room for improvement. The most significant issue is the visual 'jitter' of entities located great distances from the origin of the coordinate system, which is the Earth. Additional

enhancements could be made to the pod user interface providing an alternative means of navigation control. Furthermore, the user interface could provide information about the entity to which the user is attached. Finally, the algorithms used to determine satellite visibility and generate the random error in the VGPSR could be improved to better model actual system performance. Each of these areas of potential work is now discussed in detail.

6.2.1 Jitter Removal. The visual jitter encountered when attached to entities far from the Earth is due to software and hardware limitations. The current scale of the solar system environment results in coordinate values that exceed the maximum single digit precision floating point value of the system. However, the *Solar System Modeler* (*ObjectSim*) is forced to use the Performer pfVec structure to store the coordinates of entities to be rendered. The pfVec structure comprises three floating point values. As a result, when the coordinates of an entity exceed 2^{24}, the system rounds the value to the nearest power of two. This results in significant loss of precision for many of the coordinate values found in the *Solar System Modeler* revealed as visual jitter encountered when attached to Galileo, Pioneer 11, Voyager 2 and other planets and moons at the outer regions of the solar system.

While the next generation of hardware and software may provide double precision data structures that would eliminate the jitter problem, a more immediate solution is desirable. I recommend modifying the program to derive all players from the `DoublePlayer` class rather than the `Player` class. Then, double precision coordinates would be available for all entities. The propagation algorithms would use these values rather than the single precision floating point values to determine the positions of all entities. Following the propagation phase, the position of the pod would be translated to the origin. All other entities would then be translated by the same vector thereby maintaining the relative positions. The new coordinates would then be stored in the pfVec structures for rendering. The objects in close proximity to the viewpoint would then

have coordinate values within single precision range. Although distant objects would still suffer the same loss of precision for rendering, they would not be visible from the current viewpoint and would not jitter. Furthermore, the true position of all entities would be maintained by the double precision coordinates for the next propagation cycle.

6.2.2 Navigation Interface. The current user interface provides several parameters that may be manipulated to modify the orientation and course of the pod. While this interface is excellent for stationary observing, it is difficult to maneuver in some situations such as a flying by a satellite or planet. Consequently, a selectable interface that would allow the user to choose between the current front panel interface and an alternative front panel interface is desirable.

The alternative panel would replace the current navigation and orientation interface with two slider bars and a nine by nine grid of push buttons. The two slider bars would be controlled by the mouse and used to set forward and reverse speed. Each button in the push button grid would provide a different setting for pitch and heading. For example, the top center button would be selected for no change in heading but full downward pitch. The right center button would be selected for no pitch adjustment but for a full right hand turn. In this fashion, the grid would provide 81 discrete choices for combined heading and pitch adjustment. The grid would sense the mouse position when the left mouse button is depressed and alter the heading and pitch to correspond with the selected button. This interface would provide the ability to rapidly modify the heading, pitch, and speed of the pod during movement.

6.2.3 ABOUT Button. The *Solar System Modeler's* primary purpose is to provide a space visualization tool for a broad audience. No assumptions are made regarding the level of user knowledge about the entities in the simulation. An "ABOUT" button would provide the user a simple mechanism to obtain basic information on the satellite, planet, moon, asteroid, or comet being viewed. This button could be added to the right panel of the user interface with

minimal difficulty. The selection of information to be displayed could be tied to the interface so the data would automatically correspond to the entity the user is viewing.

6.2.4 VGPSR Models. Although the VGPSR currently generates GPS indicated positions with acceptable realism, two of the algorithms used in error generation could be improved to better model actual system operation.

First, the algorithm reused from the LCCM project to determine satellite visibility assumes ground-based receivers. To better facilitate airborne receivers, a different approach should be investigated. One such technique uses distance relationships between the center of the Earth, the receiver, and the satellite to determine visibility. This model doesn't assume the receiver is located on the Earth and would be appropriate for modeling airborne receivers. The following equation is used in a simple test to determine visibility:

$$T = \sqrt{r_e^2 + r_n^2} \tag{44}$$

where

r_e is the magnitude of the user position vector (from Earth center)
r_n is the range between the user and the satellite

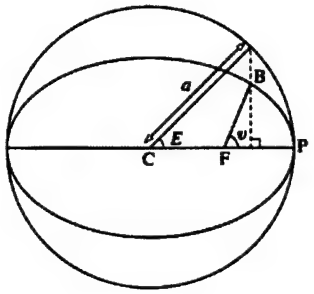
The visibility test simply evaluates T. If its value is greater than or equal to the magnitude of the satellite position vector (from Earth center), the satellite is below the horizon and not visible. Otherwise, the satellite is visible. This approach would need to be modified slightly to incorporate the mask angle.

Secondly, the algorithm used to generate the user navigation error (UNE) assumes a spherical GPS error. In fact, the error is ellipsoidal with the vertical error having greater magnitude than the horizontal error. With appropriate modifications, the current error generation algorithm could be used to generate an ellipsoidal error rather than a spherical error.

6.3 Conclusions

The *Solar System Modeler* is the first virtual environment providing a comprehensive visualization tool for space that not only facilitates exploration of the solar system, but also provides useful information to other DIS applications. Furthermore, the extensible design of the Virtual GPS Receiver provides a tool that can easily be integrated a variety of applications benefiting from virtual GPS navigation. The *Solar System Modeler* provides valuable insight to large-scale distributed virtual environments in areas such as scale, user interface design, and distributed simulation communications. Further development of this project can only lead to additional insights thereby benefiting the Air Force and the virtual environment - distributed simulation community at large.

anomaly - (27) An angle used in describing the motion of a body in an elliptical orbit. The *true anomaly, v,* is the angle between the line joining the body B to the focus of the ellipse, F, and the line joining F to periapsis, the point on the orbit closest to F (see illustration). The *mean anomaly, M*, is the angle between the line PF and the line joining F to a hypothetical body that has the same orbital period as the real one under consideration but travels at a uniform angular speed. The *eccentric anomaly, E*, is a useful parameter for expressing the variable length of the radius vector, **r**. The linking equation is $r = a(1 - e \cos E)$, where a is the semi-major axis and e the eccentricity of the elliptical orbit.



Anomaly. The geometry of true anomaly, *v*, and eccentric anomaly, *E*.
(27)

aphelion - (27) The point furthest from the Sun in the orbit of a body, such as a planet or comet, that is traveling around the Sun.

eccentric anomaly - see anomaly.

ephemeris constants - (23) A small group of parameters used in defining the orbit of a celestial body or man-made satellite.

mean anomaly - see anomaly.

obliquity of the ecliptic (symbol ε) - (27) The angle between the planes of the Earth's equator and the ecliptic (the mean plane of the Earth's orbit around the Sun).

periapsis - (5:24) The "near apse"; the point nearest the prime focus.

perihelion - (27) In orbital motion in the solar system, the point of closest approach to the Sun.

selective availability (degradation of accuracy) - (23) The process of doctoring and distorting the signals coming down from the GPS satellites so that unauthorized users cannot achieve the full military accuracy of the Navstar system.

true anomaly - see anomaly.

user navigation error (UNE) - (17) Given a sufficiently stationary and ergodic satellite constellation ranging error behavior over a minimum number of measurement intervals, multiplication, of the DOP and a constellation ranging error standard deviation value will yield an approximation of the RMS position error. The user is cautioned that any divergence away from the stationary and ergodic assumption will cause the UNE to diverge from a measured RMS value.

Appendix 1. Summary Statistics of Correlation Analysis

Standard Deviation: 1	
Mean	1.58941
Standard Error	0.02151
Median	1.55268
Mode	#N/A
Standard Deviation	0.68060
Sample Variance	0.46321
Kurtosis	-0.10659
Skewness	0.44749
Range	3.91983
Minimum	0.05786
Maximum	3.97770
Sum	1591.00
Count	1001
Confidence	0.04216

Standard Deviation: 2	
Mean	3.20782
Standard Error	0.04299
Median	3.10957
Mode	#N/A
Standard Deviation	1.36020
Sample Variance	1.85015
Kurtosis	-0.10154
Skewness	0.44172
Range	7.33887
Minimum	0.21777
Maximum	7.55665
Sum	3211.03
Count	1001
Confidence	0.08426

Standard Deviation: 3	
Mean	4.74472
Standard Error	0.06278
Median	4.56864
Mode	#N/A
Standard Deviation	1.98631
Sample Variance	3.94544
Kurtosis	-0.38998
Skewness	0.35974
Range	10.1229
Minimum	0.49885
Maximum	10.6218
Sum	4749.47
Count	1001
Confidence	0.12304

Standard Deviation: 4	
Mean	6.54067
Standard Error	0.08476
Median	6.33958
Mode	#N/A
Standard Deviation	2.68195
Sample Variance	7.19287
Kurtosis	0.77722
Skewness	0.64855
Range	19.7418
Minimum	0.80333
Maximum	20.5451
Sum	6547.21
Count	1001
Confidence	0.16614

Standard Deviation: 5	
Mean	7.81388
Standard Error	0.10506
Median	7.70711
Mode	#N/A
Standard Deviation	3.32402
Sample Variance	11.0491
Kurtosis	0.12380
Skewness	0.45389
Range	19.7593
Minimum	0.46612
Maximum	20.2254
Sum	7821.69
Count	1001
Confidence	0.20591

Standard Deviation: 6	
Mean	9.73166
Standard Error	0.13286
Median	9.27939
Mode	#N/A
Standard Deviation	4.20359
Sample Variance	17.6702
Kurtosis	-0.10025
Skewness	0.47047
Range	24.6643
Minimum	1.13414
Maximum	25.7984
Sum	9741.40
Count	1001
Confidence	0.26040

Standard Deviation: 7	
Mean	11.3804
Standard Error	0.15255
Median	10.9884
Mode	#N/A
Standard Deviation	4.82646
Sample Variance	23.2948
Kurtosis	-0.15571
Skewness	0.37170
Range	29.0045
Minimum	1.09364
Maximum	30.0982
Sum	11391.8
Count	1001
Confidence	0.29899

Standard Deviation: 8	
Mean	13.0470
Standard Error	0.17345
Median	12.8920
Mode	#N/A
Standard Deviation	5.48776
Sample Variance	30.1155
Kurtosis	0.21567
Skewness	0.44637
Range	38.0183
Minimum	1.83311
Maximum	39.8515
Sum	13060.0
Count	1001
Confidence	0.33995

Standard Deviation: 9	
Mean	14.5597
Standard Error	0.18838
Median	14.2012
Mode	#N/A
Standard Deviation	5.96027
Sample Variance	35.5248
Kurtosis	-0.05415
Skewness	0.37068
Range	36.6119
Minimum	1.40611
Maximum	38.0180
Sum	14574.3
Count	1001
Confidence	0.36922

Standard Deviation: 10	
Mean	16.2637
Standard Error	0.21701
Median	16.0939
Mode	#N/A
Standard Deviation	6.86593
Sample Variance	47.1410
Kurtosis	0.22320
Skewness	0.45461
Range	47.5229
Minimum	2.29139
Maximum	49.8143
Sum	16280.0
Count	1001
Confidence	0.42533

Standard Deviation: 11	
Mean	17.6596
Standard Error	0.23551
Median	17.0255
Mode	#N/A
Standard Deviation	7.45143
Sample Variance	55.5238
Kurtosis	-0.28586
Skewness	0.33559
Range	44.2178
Minimum	0.63991
Maximum	44.8577
Sum	17677.2
Count	1001
Confidence	0.46160

Standard Deviation: 12	
Mean	18.9450
Standard Error	0.25162
Median	18.3810
Mode	#N/A
Standard Deviation	7.96115
Sample Variance	63.3800
Kurtosis	-0.15341
Skewness	0.36939
Range	43.7027
Minimum	1.03670
Maximum	44.7394
Sum	18963.9
Count	1001
Confidence	0.49318

Standard Deviation: 13	
Mean	20.9347
Standard Error	0.27973
Median	20.4070
Mode	#N/A
Standard Deviation	8.85032
Sample Variance	78.3281
Kurtosis	0.05342
Skewness	0.44516
Range	53.9260
Minimum	1.74805
Maximum	55.6740
Sum	20955.7
Count	1001
Confidence	0.54826

Standard Deviation: 14	
Mean	22.1099
Standard Error	0.30373
Median	21.0462
Mode	#N/A
Standard Deviation	9.60961
Sample Variance	92.3447
Kurtosis	-0.17154
Skewness	0.46124
Range	55.3353
Minimum	1.22253
Maximum	56.5578
Sum	22132.0
Count	1001
Confidence	0.59530

Standard Deviation: 15	
Mean	23.7853
Standard Error	0.31563
Median	22.8035
Mode	#N/A
Standard Deviation	9.98639
Sample Variance	99.7279
Kurtosis	0.09361
Skewness	0.53216
Range	59.3684
Minimum	1.81990
Maximum	61.1883
Sum	23809.1
Count	1001
Confidence	0.61864

Standard Deviation: 16	
Mean	26.1938
Standard Error	0.36045
Median	24.9611
Mode	#N/A
Standard Deviation	11.4043
Sample Variance	130.058
Kurtosis	0.20422
Skewness	0.58160
Range	70.4096
Minimum	2.24186
Maximum	72.6515
Sum	26220.0
Count	1001
Confidence	0.70647

Appendix 2. Summary Statistics of VGPSR Error Analysis

F16D Normal Mode X Coordinate Error	
Mean	2.073768483
Standard Error	0.558814842
Median	2.454221714
Mode	#N/A
Standard Deviation	5.359963665
Sample Variance	28.72921049
Kurtosis	1.560195232
Skewness	-0.730460523
Range	30.07519662
Minimum	-15.76108675
Maximum	14.31410987
Sum	190.7867005
Count	92
Confidence Level(95.000%)	1.095255342

F16D Normal Mode Radial Error	
Mean	9.662614479
Standard Error	0.505366388
Median	8.655345002
Mode	#N/A
Standard Deviation	4.847304109
Sample Variance	23.49635713
Kurtosis	1.79816356
Skewness	1.176631922
Range	25.05117921
Minimum	3.123514738
Maximum	28.17469394
Sum	888.960532
Count	92
Confidence Level(95.000%)	0.990498453

F16D Normal Mode Y Coordinate Error	
Mean	1.387506652
Standard Error	0.617662904
Median	0.264697901
Mode	#N/A
Standard Deviation	5.92441445
Sample Variance	35.09868657
Kurtosis	3.067851772
Skewness	1.578568291
Range	31.78469062
Minimum	-6.954593558
Maximum	24.83009706
Sum	127.650612
Count	92
Confidence Level(95.000%)	1.210595253

F16D Normal Mode Z Coordinate Error	
Mean	-1.14416019
Standard Error	0.710503397
Median	-0.84138338
Mode	#N/A
Standard Deviation	6.814909173
Sample Variance	46.44298704
Kurtosis	0.001115294
Skewness	0.014464243
Range	32.85818157
Minimum	-18.1031361
Maximum	14.75504543
Sum	-105.262737
Count	92
Confidence Level(95.000%)	1.392559006

F16D Turbo X Coordinate Error	
Mean	1.855730775
Standard Error	0.49820297
Median	2.189562416
Mode	#N/A
Standard Deviation	4.778595016
Sample Variance	22.83497033
Kurtosis	1.546500039
Skewness	-0.726775816
Range	26.77310657
Minimum	-14.02755148
Maximum	12.74555509
Sum	170.7272313
Count	92
Confidence Level(95.000%)	0.976458432

F16D Turbo Radial Error	
Mean	8.592637241
Standard Error	0.442752253
Median	7.714367995
Mode	#N/A
Standard Deviation	4.246730427
Sample Variance	18.03471932
Kurtosis	1.51664906
Skewness	1.106537048
Range	21.70714023
Minimum	2.782629638
Maximum	24.48976987
Sum	790.5226261
Count	92
Confidence Level(95.000%)	0.867777186

F16D Turbo Y Coordinate Error	
Mean	1.196305157
Standard Error	0.54351692
Median	0.215182999
Mode	#N/A
Standard Deviation	5.21323116
Sample Variance	27.17777913
Kurtosis	2.849629474
Skewness	1.52489573
Range	27.81302384
Minimum	-6.230416272
Maximum	21.58260757
Sum	110.0600745
Count	92
Confidence Level(95.000%)	1.065272012

F16D Turbo Z Coordinate Error	
Mean	-1.0397904
Standard Error	0.631124281
Median	-0.74928027
Mode	#N/A
Standard Deviation	6.053531443
Sample Variance	36.64524293
Kurtosis	-0.02576038
Skewness	0.003428401
Range	29.01450011
Minimum	-16.1227548
Maximum	12.89174532
Sum	-95.660717
Count	92
Confidence Level(95.000%)	1.236979029

VC Bomb Drop X Coord. Error	
Mean	0.425799641
Standard Error	0.549366874
Median	0.443288
Mode	#N/A
Standard Deviation	8.457395527
Sample Variance	71.5275391
Kurtosis	-0.109082407
Skewness	-0.230509083
Range	45.2122
Minimum	-24.3828
Maximum	20.8294
Sum	100.914515
Count	237
Confidence Level(95.000%)	1.076737693

VC Bomb Drop Radial Error	
Mean	13.37902781
Standard Error	0.342609035
Median	13.00130919
Mode	#N/A
Standard Deviation	5.27439905
Sample Variance	27.81928534
Kurtosis	0.107847029
Skewness	0.448284542
Range	29.09681716
Minimum	3.718445391
Maximum	32.81526255
Sum	3170.829591
Count	237
Confidence Level(95.000%)	0.671500375

VC Bomb Drop Y Coord. Error	
Mean	2.322750723
Standard Error	0.513497351
Median	1.45914
Mode	#N/A
Standard Deviation	7.905191238
Sample Variance	62.49204851
Kurtosis	-0.056967313
Skewness	0.477135958
Range	43.7814
Minimum	-14.8616
Maximum	28.9198
Sum	550.4919214
Count	237
Confidence Level(95.000%)	1.006434824

VC Bomb Drop Z Coordinate Error	
Mean	-1.56957514
Standard Error	0.525639816
Median	-1.04467
Mode	#N/A
Standard Deviation	8.092122109
Sample Variance	65.48244022
Kurtosis	-0.36171705
Skewness	0.051145197
Range	38.8782
Minimum	-21.6038
Maximum	17.2744
Sum	-371.989308
Count	237
Confidence Level(95.000%)	1.030233583

Bibliography

1. Akely, Kurt. "Reality Engine Graphics," *SIGGRAPH 93 Conference Proceedings,* 109-116 (1993).

2. Ananda, Mohan. "The Global Positioning System (GPS) Accuracy, System Error Budget, Space and Control Segment Overview," *Agard Lecture Series No. 161, The NAVSTAR GPS System*, 4-1 through 4-17 (October 1988).

3. Ananda, Mohan. "The Global Positioning System (GPS) Constellation and Coverage," *Agard Lecture Series No. 161, The NAVSTAR GPS System*, 3-1 through 3-8 (October 1988).

4. Banks, J., and others. *Discrete-Event Simulation*, Prentice Hall, Upper Saddle River, 1996.

5. Bate, R.R., and others. *Fundamentals of Astrodynamics*, Dover Publications, Inc., New York, 1971.

6. Bergman, L. D., and others. "View - An Exploratory Molecular Visualization System with User-Definable Interaction Sequences." *SIGGRAPH 93 Conference Proceedings*, 117-126 (1993).

7. Blau, B., and others. "The DIS (Distributed Interactive Simulation) Protocols and their Application to Virtual Environments," Institute for Simulation and Training, Orlando, FL, 1994.

8. Cater, J.P. and S. D. Huffman. "Use of the Remote Access Virtual Environment Network (RAVEN) for Coordinated IVA-EVA Astronaut Training and Evaluation," *Presence*, 103-109 (Spring 1995).

9. Crampton, G.H. (editor). *Motion and Space Sickness*, CRC Press, Boca Raton, 1990.

10. DeVilbiss, Stewart. Faculty, Air Force Institute of Technology, WPAFB OH. e-mail: sdevilbi@afit.af.mil. Personal interview. October 1996.

11. DMA Technical Report, *Department of Defense World Geodetic System 1984: Its Definition and Relationship with Local Geodetic Systems*, DMA TR 8350.2, 30 September 1987.

12. Easton, R.L. "The Navigation Technology Program," *Journal of the Institute of Navigation*, Vol. 25, No. 2, (1978).

13. Ellis, S.R. "Nature and Origins of Virtual Environments: A Bibliographical Essay," *Computing Systems in Engineering*, 321-347 (2(4) 1991).

14. Ellis, S.R. *Pictorial Communication in Virtual and Real Environments*, Taylor and Francis. London, 1991.

15. Gossweiler, and others. "An Introductory Tutorial for Developing Multi-user Virtual Environments," *Presence*, 255-264 (Vol. 3 Num. 4, 1994).

16. Global Positioning System Standard Positioning Service Signal Specification, Annex A, June 1995.

17. IEEE. "IEEE Standard for Information Technology - Protocols for Distributed Interactive Simulation Applications," Institute of Electrical and Electronics Engineers, Inc., IEEE1278-1993, Piscataway, NJ, 1993.

18. IEEE. "Standard for Distributed Interactive Simulation--Application Protocols," Institute of Electrical and Electronics Engineers, Inc., IEEE1278.1-1995, Piscataway, NJ, 1995.

19. IST. "Enumeration and Bit Encoded Values for Use with Protocols for Distributed Interactive Simulation Applications," Institute for Simulation and Training, IST-CR-96-06, Contract #N61339-94-C-0080, Orlando FL, 1996.

20. IST. "Standard for Distributed Interactive Simulation - Application Protocols, Version 2.0 Fourth Draft," Institute for Simulation and Training, Contract #N661339-91-C-0091, Orlando, FL, 1994.

21. Kelso, T.S. *Celestial WWW: Celestial BBS on the Internet*, http://www.grove.net/~tkselso/

22. Kunz, Andrea. *A Virtual Environment for Satellite Modeling and Orbital Analysis in a Distributed Interactive Simulation*. MS thesis, AFIT/GCS/ENG/93D-14, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.

23. Logsdon, Tom. *The Navstar Global Positioning System*, Van Nostrand Reinhold, New York, 1992.

24. McCabe, James D. "Long Haul Communications Support for Visualization of Computational Aeroscience Simulations." *Presence*, 110-120 (Spring 1995).

25. McKinnon, G.M. and R. Kruk. "Multiaxis Control of Telemanipulators," *Pictorial Communication in Virtual and Real Environments*, Taylor and Francis, London, 1991.

26. Meeus, Jean. *Astronomical Algorithms*, Willmann-Bell Inc., Richmond, 1991.

27. Mitton, Jacqueline. *A Concise Dictionary of Astronomy*, Oxford University Press, New York, 1991.

28. Nieuwejaar, Per W. "GPS Signal Structure," *Agard Lecture Series No. 161, The NAVSTAR GPS System*, 5-1 through 5-6 (Oct 1988).

29. Moses, R.A. and W. M. Hart *Adler's Physiology of the Eye*, 8th Edition, Mosby, Washington D.C., 1987.

30. National Research Council, edited by Durlach, N.I. and Mavor, A.S. *Virtual Reality: Scientific and Technological Challenges*, National Academy Press, Washington D.C., 1995.

31. Pearson, R. and M. Qiu. *Development of a Simple Virtual Prototype for the Low-Cost Competent Munition*, US Army Research Laboratory, Aberdeen Proving Ground, MD.

32. Rohlf, John and James Helman. "IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics," *SIGGRAPH 94 Conference Proceedings*, 1994.

33. Roy, Trina M. "Cosmic Worm in the CAVE: Steering a High-Performance Computing Application from a Virtual Environment," *Presence*, 121-129 (Spring 1995).

34. Seidelmann, P.K. (editor), *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Mill Valley CA, 1992.

35. Sherman, William R., and Alan B. Craig. "Literacy in Virtual Reality: a New Medium," *Computer Graphics*, 37-41 (November 1995).

36. Smith, R.S., and others. "Dynamic spacecraft simulators in Military Space Ground Systems," *Proceedings of the Symposium on Military Space Communications and Operations*, 119-126 (1983).

37. Snyder, Mark. *ObjectSim Application Developers Manual.* Air Force Institute of Technology, WPAFB, OH, 1993.

38. Stytz, Martin. "Distributed Virtual Environments," *IEEE Computer Graphics and Applications*, 19-31 (May 1996).

39. Stytz, Martin and others. "Portraying and Understanding Large-Scale Distributed Virtual Environments: Experience and Tentative Conclusions." *Presence*, 146-168 (Vol. 4 Num. 2, 1995).

40. Tokai, Shogo and others. "A Method Of Expression of Space Phenomena with CG Animation: the Collision of the Comet Shoemaker-Levy 9 with Jupiter in July 1994." *Computer Graphics: Developments in Virtual Environments*, 267-280, Academic Press Ltd., San Diego, 1995.

41. *U.S. Coast Guard Navigation Center*, http://www.navcen.uscg.mil/

42. Vanderburgh, J.C. *Space Modeler: An Expanded, Distributed, Virtual Environment for Space Visualization*, MS thesis, GCS/ENG/94D-23, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1994.

43. Vince, John. *Virtual Reality Systems*, Addison-Wesley, Cambridge, 1995.

Vita

Captain Gary E. Williams was born on 18 December 1961. He graduated from Calvary Christian School, Taylorsville, KY in 1980. He enlisted in the U.S. Air Force in 1981 as an electronic technician. In 1988, he was selected for the Airman Education and Commissioning Program (AECP) and was reassigned to Wright State University to complete a Bachelor of Science degree in Computer Science. He graduated Suma Cum Laude in 1991. He then served as a customer support representative and computer graphics software engineer at the National Air Intelligence Center. His next assignment was to the Air Force Institute of Technology (AFIT) where he was a technology integration team leader. He entered the School of Engineering, AFIT in May 1995.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1996	Master's Thesis

4. TITLE AND SUBTITLE

SOLAR SYSTEM MODELER: A DISTRIBUTED, VIRTUAL ENVIRONMENT FOR SPACE VISUALIZATION AND GPS NAVIGATION

5. FUNDING NUMBERS

6. AUTHOR(S)

Gary E. Williams

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/96D-29

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Capt Mark Snyder
Phillips Laboratory/VTS
3550 Aberdeen SE
Kirtland AFB, NM 87117-5776

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The *Solar System Modeler* (SM) extends the *Space Modeler* developed in 1994. It provides a virtual environment enabling an explorer to dynamically investigate near Earth satellites, deep space probes, planets, moons, and other celestial phenomena. The explorer navigates the virtual environment via mouse selected options from menu panels while wearing a tracked, head mounted display (HMD). Alternatively, a monitor may replace the HMD and keyboard controls replace head tracking. The SM's functionality is extended by the ability to broadcast simulated GPS satellite transmissions in compliance with Distributed Interactive Simulation (DIS) protocol standards. The transmissions include information found in true GPS broadcasts that is required for a receiver to determine its location. The Virtual GPS Receiver (VGPSR) receives the GPS transmissions from the SM and computes the receiver's position with a realistic error based on numerous variables simulating those encountered in the real GPS system. The VGPSR is designed as a plug-in module for simulations requiring virtual navigation. The receiver's client application provides the VGPSR with the simulation time and the true position of the receiver. In return, the application receives a GPS indicated position.

14. SUBJECT TERMS

Virtual Environments, Space Visualization, Orbital Mechanics, Distributed Interactive Simulation, DIS, Global Positioning System, GPS

15. NUMBER OF PAGES

87

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

★U.S.GPO:1993-0-336-043